Instituto Tecnológico Autónomo de México

Análisis de autoría de textos basado en técnicas de computación suave

TESIS

QUE PARA OBTENER EL TÍTULO DE

Licenciado en Matemáticas Aplicadas

PRESENTA

Javier Sagastuy Breña

ASESOR

Dr. Ángel Fernando Kuri Morales

REVISOR

Dr. Edgar Possani Espinosa

CIUDAD DE MÉXICO 2017

"Con fundamento en los artículos 21 y 27 de la Ley Federal de Derecho de Autor y como titular de los derechos moral y patrimonial de la obra titulada "**Análisis de autoría de textos basado en técnicas de computación suave**", otorgo de manera gratuita y permanente al Instituto Tecnológico Autónomo de México y a la biblioteca Raúl Baillères Jr., autorización para que fijen la obra en cualquier medio, incluido el electrónico, y la divulguen entre sus usuarios, profesores, estudiantes o terceras personas, sin que pueda percibir por tal divulgación una prestación"

Javier Sagastuy Breña

_____

Fecha

_____

Firma

A mi mamá, por su eterno amor
y cariño incondicional.
A mi papá, por ser mi faro cuando
navego a la deriva.
A mi hermana, por su incansable sinceridad
y apoyo en los momentos más oscuros.
A mi hermano, por incitarme a
siempre ser mejor.
A mis abuelos, por inspirarme y mostrarme que
"no hay camino, se hace camino al andar".

Quisiera agradecer a mi asesor, el Dr. Ángel Kuri,
por sus ideas, sus retos y su tiempo.

Agradezco también a mis sinodales, el Dr. Fernando Esponda,
el Dr. Octavio Gutiérrez, el Dr. Ernesto Barrios y a mi revisor
el Dr. Edgar Possani por su apoyo, por el conocimiento transmitido
durante sus clases, y por su impecable revisión de este trabajo.

También quiero agradecer a la Dra. Ana Lidia Franzoni por
acompañarme desde el primer momento que pisé el ITAM.

A las secretarias administrativas, Carmen Mejía y Trinidad Martínez
por su paciencia al guiarme por todos los trámites del ITAM.

Agradezco, por último, a todos ustedes, que me
acompañaron a lo largo de mi carrera y con quienes
comparto entrañables recuerdos de esta etapa de mi vida.

Sin ustedes, no hubiera sido lo mismo.

¡Gracias!

Resumen

En este trabajo se busca resolver el problema de atribución y verificación de autoría de textos. Se diseña un sistema capaz de medir la similitud entre dos textos que consta de tres fases. Primero, se procesa el texto original y se modela a través de una base de datos estructurada con variables tanto numéricas (variables cuantitativas) como categóricas (variables cualitativas nominales). Se hizo un estudio sobre rasgos lingüísticos que permiten distinguir entre autores en el pasado y con base en ellos se definieron las variables categóricas presentes en el modelo del texto. En segundo lugar, la representación del texto como una base de datos estructurada se codifica numéricamente con un algoritmo novedoso basado en redes neuronales y algoritmos genéticos. Por último, se analizan las bases de datos numéricas resultantes para detectar similitud entre autores. Para ello se diseñó un algoritmo para detectar el estilo de escritura del autor y se definieron las métricas de similitud. Los resultados del sistema propuesto al correrse con seis conjuntos de datos distintos no indicaron que fuera posible discernir entre autores distintos.

Palabras clave: análisis de autoría, minería de textos, redes neuronales, algoritmos genéticos

Abstract

In this work, we seek to solve the problem of text attribution and verification. We designed a three-component system to measure similarity between two given texts. The first component creates a structured database with both qualitative and quantitative variables which models a given text. We based our choice of variables on the linguistic features used successfully to identify authorship on previous published research. The second component encodes the representation of the texts to be analyzed into a purely numerical database by means of a novel algorithm based on neural networks and genetic algorithms. The third and last component analyzes the resulting numerical databases to measure similarity between texts. To accomplish this, we designed an algorithm to identify an author's writing style as well as the metrics needed to measure similarity. Our results on six datasets show no evidence that our approach can effectively make a distinction between different authors.

Keywords: authorship analysis, text mining, neural networks, genetic algorithms

Índice general

Índice de figuras

Índice de tablas

1. Introducción

> *[...] vinieron los expertos y los suspicaces a decir que el poema era apócrifo, que el poema no era de Jorge Luis Borges [...] La cosa me extrañaba, pero poco me importaba. No me preocupaba mucho el problema de su autoría: el soneto era hermoso, el soneto era importante para mí, y eso era suficiente.*
>
> – Héctor Abad Faciolince

Este trabajo discute la identificación del estilo de escritura de un autor a través de un análisis fundamentalmente numérico. Al utilizar un algoritmo novedoso de codificación de variables categóricas[1] , se busca transformar el problema original rápidamente en uno numérico que pueda ser resuelto con herramientas ya existentes. La idea principal es que la metodología propuesta para resolver el problema numérico sea lo suficientemente general para que pueda ser reutilizada para resolver problemas de identificación de estilo en obras musicales o de pintura.

En este capítulo se plantea el problema a resolver, se exponen los objetivos y el alcance de este proyecto de tesis y se ilustra la metodología seguida partiendo de la forma en la que está estructurado este documento. Por último, se justifica este trabajo ilustrando posibles aplicaciones que podría tener.

[1] En este trabajo por "variable categórica" se deberá entender una variable cualitativa nominal. La codificación de una variable categórica hace referencia a la sustitución de sus valores nominales por códigos numéricos. Es importante resaltar que aunque en este trabajo se van a utilizar variables cualtitativas nominales, el algoritmo de codificación es también aplicable a variables cualitativas ordinales.

1.1. Planteamiento del problema

En este trabajo se busca diseñar un sistema que pueda identificar si dos textos fueron escritos por el mismo autor. El reto fundamental es resolver este problema con un enfoque general, puramente numérico, de tal forma que el núcleo de este trabajo sea fácilmente aplicable a otras disciplinas como música o pintura.

1.2. Objetivos y alcance

En relación al planteamiento del problema, se pueden distinguir tres grandes retos a resolver en este trabajo. Primeramente será necesario identificar las características lingüísticas que podrían permitir distinguir entre un autor y otro. En segundo lugar, será necesario modelar esta información de tal forma que se le pueda alimentar al algoritmo de codificación categórica y se produzca una codificación útil. Por último, habrá que definir un marco de procesamiento numérico que permita medir similitud entre las bases de datos suficientemente general para poderse usar más allá del análisis de textos. Con base en esto, se pueden definir los siguientes objetivos:

- Definir una estructura basada en características que permitan distinguir entre autores para modelar textos, información de carácter no-estructurado.
- Parametrizar el algoritmo de codificación categórica para obtener datos numéricos que preserven los patrones presentes en el modelo del texto.
- Diseñar un algoritmo de procesamiento numérico que permita medir similitud entre dos modelos de textos y que pueda ser utilizado para distinguir entre autores. Dado que los modelos de textos son puramente numéricos, dicho algoritmo debe ser suficientemente general para poder ser aplicado a otro tipo de datos.
- Identificar de forma abstracta el estilo de escritura de un autor, de tal forma que se pueda entrenar un sistema para detectarlo y medir la similitud de nuevos textos contra el estilo general de un autor conocido.

El alcance de este trabajo será explorar las posibilidades de análisis que ofrece el algoritmo de codificación categórica en el problema particular de análisis de autoría en textos. Se realizará un análisis lingüístico de características que puedan ser utilizadas para discernir entre autores. Con base en ese análisis se definirá la estructura que

tendrá el modelo de textos que se alimentará al algoritmo de codificación categórica. Dicha estructura permitirá modelar el texto como una base de datos con variables tanto numéricas como categóricas. Se desarrollará un sistema de preprocesamiento del texto que permita a partir del análisis realizado anteriormente, encajonar los textos en lenguaje natural dentro de una estructura rígida. Tras un análisis para encontrar la mejor parametrización del algoritmo de codificación categórica, se obtendrá la base de datos puramente numérica resultante de codificar las variables categóricas con números. Se definirá un algoritmo de procesamiento numérico para medir similitud entre dos bases de datos numéricas. Por último, se pondrá a prueba todo el esquema propuesto sobre un conjunto de textos en español para verificar que el sistema es en efecto capaz de distinguir entre autores.

La tesis analizará algunas posibilidades para solucionar cada uno de los retos descritos anteriormente, pues hacer un análisis exhaustivo de todas ellas resulta infactible. Sin embargo, se dejarán claramente indicadas las suposiciones y decisiones de diseño que podrían modificarse en trabajos futuros. En esta tesis no se entrenará un sistema que dado un texto identifique a cuál de un conjunto de estilos conocidos se asemeje más, pues para ello se depende de que el sistema sea capaz de discernir entre autores. De funcionar, ésta sería la primera aplicación inmediata que puede tener este trabajo.

1.3. Metodología y estructura

Para mejor ilustrar la metodología que se seguirá, se da una breve descripción de la estructura de este trabajo en términos de los capítulos que lo componen. En el capítulo 2 se expone trabajo previo en análisis de estilos de escritura y se identifican características lingüísticas que han sido utilizadas con éxito para identificar autoría. En el capítulo 3 se exponen los fundamentos matemáticos de las principales herramientas que se utilizarán en este trabajo. En el capítlo 4 se explica el flujo de información en esta propuesta. Se justifica cada uno de los componentes y herramientas utilizadas. En el capítulo 5 se describen los experimentos que validarán el funcionamiento del sistema, así como los conjuntos de datos a utilizar. Se comentan y se analizan los resultados obtenidos. Por último, en el capítulo 6 se comenta el proyecto de tesis en general. Se exponen algunas aplicaciones y se identifican claramente líneas de trabajo futuro.

1.4. Justificación

Un sistema capaz de hacer lo que se plantea en esta tesis resultaría extremadamente útil para verificar que el supuesto autor de un texto sea en efecto el autor verdadero. Con esto, un sistema así, podría ser capaz de detectar plagio. No sólo eso, al poder generar una representación del estilo de escritura de un autor, si esto se hace en textos sucesivos, se pueden rastrear los cambios en el estilo de escritura de varios autores a través del tiempo. Del mismo modo, dado un catálogo de "firmas estilística" podemos identificar intentos de falsificación de textos.

Dado que el algoritmo de codificación categórica se puede aplicar a bases de datos de todo tipo, las aplicaciones de este trabajo no tienen por qué limitarse a identificar la autoría de textos. Si se cumple con el objetivo de diseñar un algoritmo numérico lo suficientemente general, se abre una amplia gama de aplicaciones. Se le podrían dar como entrada composiciones musicales, y podría ser posible atribuir la autoría de dichas piezas. Si se le dieran como entrada obras de pintura, se podrían detectar falsificaciones de obras clásicas. De este modo, el enfoque aquí propuesto resulta verdaderamente poderoso para detectar patrones complejos que quizás un ser humano puede encontrar con facilidad (por ejemplo, alguien que ha leído a Borges quizás podría identificar fácilmente un texto de Borges que nunca antes haya leído), pero que no es trivial detectar computacionalmente.

2. Estilometría

> *Ya no creo que el actor William Shakespeare de Stratford sea el autor de las obras que durante tanto tiempo se le atribuyeron.*
>
> – Sigmund Freud

La estilometría se refiere al estudio estadístico del estilo con el que una persona produce cierta obra. Usualmente se trabaja con estilos lingüísticos y se concentra en el estudio de lenguaje escrito. Sin embargo, es un campo general que ha llegado a ser aplicado a música y a pintura.

La hipótesis fundamental sobre la que se basa la estilometría es que el ser humano, al tener hábitos fuertemente arraigados, le imprime de manera inconsciente el mismo estilo a sus obras como si fuese una huella dactilar.

2.1. Aspectos generales

2.1.1. El problema de análisis de autoría

El análisis de autoría es una vertiente de la estilometría aplicada a lenguaje escrito. Es un área que ha tenido creciente interés en los últimos años. De acuerdo a Brocado [4], existen tres vertientes del análisis de autoría: atribución, verificación y perfilado de autoría.

La *atribución de autoría* busca encontrar el autor al que con mayor probabilidad se le puede atribuir una obra, dentro de una lista de individuos conocidos.

La *verificación de autoría* pretende comprobar si un documento, en efecto fue escrito por el autor al que se le atribuye.

El *perfilado de autoría* consiste en la caracterización de un documento anónimo con

base en diversas características del autor, como pueden ser género, edad y raza.

Este trabajo se centra alrededor de la atribución y la verificación de autoría, enfocándose fundamentalmente en la primera vertiente. Además, en lo que respecta a la verificación de autoría, hay poca literatura al respecto más allá de temas como detección de plagio.

2.2. Trabajo previo

Entre las primeras personas que utilizaron herramientas de estadística multivariada para analizar textos se encuentran Ledger y Merriam [5] y Holmes y Forsyth [6]. El enfoque de Ledger y Merriam utiliza agrupamiento de mínima varianza de Ward, análisis de componentes principales y análisis de discriminante con marcadores léxicos similares a los que se explican en la sección 2.3. Mencionan que este enfoque no es útil para muestras de texto de menos de 500 palabras y recomiendan que se tomen textos mucho más largos. Además, el método no escala bien cuando se incrementa el número de autores analizados. En este artículo se analiza la obra *The Two Noble Kinsmen* atribuida a John Fletcher y William Shakespeare en conjunto y busca determinar qué partes de la obra se asemejan más al estilo de escritura de cada autor. Holmes y Forsyth utilizan agrupamiento, análisis de componentes principales y un algoritmo genético para encontrar reglas de atribución de autoría. Trabajan con el famoso problema de los 77 artículos de *The Federalist* publicados en 1787 y 1788 bajo el pseudónimo de *Publius*. Poco después de su publicación, tres autores se atribuyeron la autoría de diferentes artículos. En doce de ellos, más de un autor decía haberlo escrito y por ello, es un problema muy interesante de atribución de autoría.

Más adelante, Peng [7] también emplea técnicas de estadística multivariada como análisis de componentes principales y análisis de discriminante a textos de diversos autores angloparlantes. Entre éstos destacan Shakespeare y su contemporáneo Marlowe. El debate de autoría de los textos de Shakespeare ha sido ampliamente estudiado con un gran número de enfoques y expertos en el tema tienen opiniones muy divididas al respecto.

En trabajos posteriores se analizan textos más cortos buscando obtener una mejor precisión para poder distinguir entre un mayor número de autores. Por ejemplo, Abbasi y Chen [8] proponen una técnica para identificar lo que ellos llaman "writeprint", la huella de escritura de un autor. Basándose en ella, diferentes estudios han atacado el

problema de atribución de autoría en correos electrónicos y medios digitales, donde la disponibilidad de textos no es tan elevada, y éstos son relativamente cortos. Ejemplos son los trabajos de Iqbal [9, 10], Abbasi et. al. [8], Chen [11], de Vel [12] y Brocardo [4]. Abbasi y Chen [8] utilizan transformadas de Karhunen-Loeve, una variante supervisada de análisis de componentes principales, para definir las huellas de escritura. Con ellas, logran precisiones de clasificación de hasta 94 % al trabajar con 100 autores distintos. Otra idea que llama la atención de su enfoque es que mencionan que no solo la presencia de un conjunto de características es determinante del estilo de escritura de un autor, también la ausencia de cierta característica puede ser altamente determinante. Esta idea ha sido ampliamente desarrollada por El Dr. Fernando Esponda en sus trabajos sobre bases de datos negativas [13] en las cuales se almacena todo menos la información de interés.

Iqbal [10] utiliza un enfoque que le permite extraer de cada texto un vector de características estilísticas. Esto es, se miden características como las que se exponen en la sección 2.3 y se agrupan en un vector (o lista) que resulta ser una representación simplificada del texto. Para todos los vectores generados por los textos de un mismo autor, aplica un algoritmo de agrupamiento para acercarse a la huella de escritura del autor. A cada cúmulo de vectores le extrae los patrones más frecuentes y termina con una lista de características estilísticas que conforman una interpretación intuitiva de la huella de escritura del autor. Esto último constituye la gran ventaja de su método frente a otros: el estilo del autor se puede expresar en términos de características bien definidas, mientras que en un gran número de otros enfoques (como el que se propone en este trabajo) la representación del estilo es abstracta.

Estudios de los últimos años han atacado el problema de atribución de autoría utilizando enfoques innovadores y extremadamente interesantes. Qian y Li [14] en lugar de trabajar en el espacio de todos los documentos a analizar, transforman dicho espacio a un espacio que ellos llaman de similitud. En resumen, en lugar de trabajar directamente sobre el conjunto de documentos (o representaciones de documentos) $D = \{d_1, d_2, ..., d_n\}$, generan el espacio de similitud a partir de D. Dicho espacio se puede pensar de forma simplificada como algo parecido a $S = \{s(d_i, d_j) | d_i, d_j \in D, i \neq j\}$ en donde la función s es una forma de medir similitud entre documentos. El detalle de la transformación de espacios se puede consultar en el artículo original. Su propuesta nuevamente identifica de forma abstracta el estilo de escritura y utiliza un umbral de similitud para atri-

buir una obra nueva a un autor conocido. Ferrilli [15] propone describir las relaciones estructurales en los textos analizando utilizando lógica de primer orden. Tras definir una medida de similitud entre estas representaciones relacionales, se puede aplicar un algoritmo de agrupamiento. Los cúmulos resultantes modelan el estilo de escritura del autor. Al usar agrupamiento jerárquico aglomerativo, es sencillo comparar el nivel de similitud entre dos grupos de cúmulos resultantes de analizar dos textos independientes. Los resultados presentados por Ferrilli son buenos para algunos conjuntos de datos y bastante malos para otros. Sin embargo, las ideas que presenta son interesantes y tienen potencial para ser exploradas en el futuro.

2.3. Algunos marcadores estilísticos relevantes

Estudios de lingüística han identificado cientos de características que han probado ser efectivas para identificar autoría. En este trabajo se hará referencia a ellas o grupos de ellas con el término **marcadores estilísticos**, pues definen precisamente los rasgos distintivos del estilo de escritura de un autor. Combinaciones de ellos se han empleado en conjunto con diferentes técnicas estadísticas de aprendizaje tanto supervisado como no supervisado para abordar este problema. Los métodos de aprendizaje supervisado son la herramienta apropiada para el problema de atribución de autoría cuando se cuenta con textos previos de los cuales se pueda extraer el estilo de escritura de un autor. Sin embargo, cuando no se dispone de ejemplos anteriores y se busca medir similitud entre estilos, se pueden utilizar métodos no supervisados (como agrupamiento).

Chen e Iqbal [11, 10] exponen un amplio conjunto de marcadores estilísticos que se han utilizado con éxito en el pasado. De acuerdo a Chen, éstos se pueden dividir en cuatro categorías: léxicos, sintácticos, estructurales y específicos al contenido.

Los marcadores **léxicos** son de bajo nivel, esto es, se encuentran a nivel de palabra e involucran los caracteres utilizados y características propias de la palabra aislada. Ejemplos de éstos son: frecuencia de letra, longitud de palabra y número de palabras por oración. Estudios como el de Ledger y Merriam [5] emplean únicamente marcadores de este estilo y obtienen buenos resultados.

Los marcadores **sintácticos** se encuentran a nivel de oración. Incluyen signos de puntuación y palabras funcionales (artículos, preposiciones, conjunciones y pronombres).

Dichos elementos, si bien tienen poco significado léxico, ayudan a capturar relaciones gramaticales en una oración. En esta categoría también están incluidas medidas que involucran las categorías y subcategorías gramaticales de las palabras. Esto se mide utilizando un conjunto de etiquetas estándar llamadas **POS tags** por su nombre en inglés ("Part of Speech tags"). De Vel [12] obtuvo resultados prometedores empleando marcadores de este tipo.

Los marcadores **estructurales** permiten capturar información a nivel de documento y analizan la disposición y organización del texto. Ejemplos de ellos son longitud de párrafo, presencia de líneas en blanco y número de párrafos. Son especialmente útiles cuando se analizan cartas o correos electrónicos, pues el autor le imprime con más certidumbre una estructura propia al texto. Sin embargo, la longitud de párrafo resulta útil para analizar prosa más extensa.

Por último, los marcadores **específicos al contenido** son simplemente colecciones de palabras clave o frases importantes relevantes para un tema determinado. Por ejemplo, se puede detectar la presencia de un sentimiento, analizando si existen palabras que normalmente se asocian con dicho sentimiento. También se suelen usar $n-$gramas: secuencias predefinidas de n palabras. Por lo general, se analiza la frecuencia con la que éstas aparecen en el texto y deben ser definidas con anticipación.

Los estudios más recientes y sofisticados emplean una amplia mezcla de los marcadores antes mencionadas. Un ejemplo es el de Chen [11] el cuál emplea 120 marcadores de todas las categorías. En la tabla 2.1 se presenta un resumen de algunos de los marcadores estilísticos que aparecen en los estudios mencionados anteriormente.

Con base en estos estudios y características detectadas, se define en el capítulo 4 el modelo que se utilizó para capturar la huella estilística del autor de un texto.

Léxicos	**Sintácticos**
Número de caracteres (ch)	Número de signos de puntuación
Número de letras/ch	Número de signos de puntuación/ch
Número de mayúsculas/ch	Frecuencia de POS tags
Número de dígitos/ch	Proporciones de POS tags
Longitud promedio de palabra	$n-$gramas de POS tags
Número de tokens	Número de palabras funcionales
Longitud promedio de oración	Proporciones de palabras funcionales
Estructurales	**Específicos al contenido**
Saludos, despedidas	Sentimientos
Espacios en blanco	$n-$gramas de palabras
Número de párrafos	
Número de palabras por párrafo	
Número de oraciones	
Número de líneas	
Longitud promedio de línea	

Tabla 2.1. Algunos marcadores estilísticos

3. Fundamentos Teóricos

> *If our brains were simple enough for us to understand them, we'd be so simple that we couldn't.*
>
> – Ian Stewart

El algoritmo desarrollado en este trabajo se basa en diferentes herramientas para extraer los patrones presentes en un texto, definir de forma abstracta la huella de escritura de un autor y establecer similitud entre huellas.

En este capítulo se exponen los fundamentos matemáticos de los diferentes componentes que intervienen en esta propuesta.

3.1. Algoritmos genéticos

Cuando se habla de un problema de optimización, se pretende resolver el problema general

$$\begin{aligned} &\text{mín } f(x) \\ &\text{s.a. } x \in \mathcal{D} \end{aligned} \tag{3.1}$$

Donde $f : \mathbb{R}^n \to \mathbb{R}$ y $\mathcal{D} \subset \mathbb{R}^n$ es la región factible. La definición anterior es equivalente a la que presenta Nocedal [16] si se supone que $\mathcal{D}$ se puede definir como el conjunto de puntos que cumplen una serie de condiciones que pueden ser tanto de igualdad como de desigualdad. Ambos, tanto la región factible como la función pueden ser tan complejos como se desee. Normalmente, los problemas de optimización se categorizan con base en las características de f y de $\mathcal{D}$. Para las diferentes categorías de problemas, existen diferentes métodos eficientes para resolverlos. Por ejemplo: un problema con

función objetivo lineal y restricciones lineales pertenece a la categoría de los problemas de programación lineal y se pueden emplear varios métodos como el método Simplex de Dantzig para resolverlo.

A grandes rasgos, los algoritmos que se usan para resolver problemas de optimización se pueden dividir, como expone Andrea Ibarra en su tesis [17], en dos grandes grupos: exactos y metaheurísticas. Los algoritmos exactos tienen una garantía de convergencia, pues exploran el espacio de soluciones aprovechando la estructura del problema. Si además se usara un algoritmo exacto determinístico, se exploraría el espacio de soluciones en una sucesión que convergerá a la solución. Dado que existe una infinidad de problemas para los cuales la cardinalidad del espacio de soluciones o la estructura misma del problema impide encontrar la solución eficientemente, se recurre a métodos 'no exactos' como las metaheurísticas.

3.1.1. Metaheurísticas y optimización

Conforme se incrementa la complejidad de la función objetivo, de las restricciones y la dimensionalidad del problema, los costos computacionales de emplear los algoritmos matemáticos exactos para resolver estos problemas se vuelven elevados y resulta difícil obtener resultados en tiempo razonable. Es aquí en donde entran las metaheurísticas. Sörensen [18] las define como una descripción de alto nivel de un conjunto de estrategias para desarrollar algoritmos de optimización heurísticos, independientemente del problema a resolver. También menciona que recientemente se ha usado el término para referirse a los heurísticos específicos que resultan de tal descripción, aunque precisa que las metaheurísticas son puramente las ideas, conceptos y operadores que se pueden usar para diseñar un algoritmo heurístico de optimización. Por ejemplo, los algoritmos genéticos son una metaheurística para la que se han definido numerosas implementaciones específicas para distintos tipos de problemas. Aunque en sentido estricto, no existe una teoría rigurosa de convergencia para las metaheurísticas, se ha visto que en la práctica funcionan muy bien. A pesar de ser en ocasiones computacionalmente intensivas, muchas veces resulta más eficiente resolver un problema tan complejo utilizando una metaheurística que un método exacto.

Existe una amplia variedad y diferentes taxonomías para las metaheurísticas. El lector interesado puede consultar un análisis comparativo de tres de ellas en la tesis de Andrea

Ibarra [17]. En particular, aquí se analizarán únicamente los algoritmos genéticos, por ser la metaheurística que se emplea en el algoritmo central de este trabajo: *Categorical Encoding with Neural Networks and Genetic Algorithms*[19] (abreviado como CENG de ahora en adelante). En la sección 4.4 se profundizará en el funcionamiento del CENG.

3.1.2. Características generales de los algoritmos genéticos

El algoritmo genético canónico (CGA por sus siglas en inglés) fue propuesto por Holland en 1975 [20].

Definición 3.1. Un **algoritmo genético** es un algoritmo que tiene las siguientes características:

1. Opera en un espacio discreto de n dimensiones $\mathcal{D} \subset \mathbb{N}^n$, en lugar de en $\mathbb{R}^n$.
2. Explora el espacio de soluciones $\mathcal{D}$ buscando una aproximación al vector x^* solución del problema (3.1) analizando simultáneamente un conjunto finito de soluciones candidatas $S(t) \subset \mathcal{D}$ (individuos de la población) al tiempo (o en la generación) t, $t \in \mathbb{N}$.
3. Los elementos de $S(t) = \{s_1(t), s_2(t), ..., s_m(t)\}$ están codificados de alguna forma apropiada (genes), donde s_i es la solución candidata i o individuo en la población $S(t)$ en la t-ésima generación y m es el tamaño de la población.
4. La información referente a la aptitud de los elementos de $S(t)$ se obtiene al evaluar la función objetivo del problema (3.1) $\forall s_i(t) \in S(t)$.
5. Los elementos de $S(t)$, ahora calificados, se analizan para seleccionar un subconjunto apropiado para mejorar la búsqueda de la solución en $\mathcal{D}$ al cruzarlos (selección).
6. Para los elementos seleccionados, se combinan secciones de sus códigos (cruza).
7. Ciertos símbolos de los códigos de los elementos de $S(t)$ se alteran aleatoriamente (mutación).
8. Se preserva un subconjunto con los mejores elementos de $S(t)$ y se descartan el resto de ellos (elitismo / selección natural).
9. Se regresa al paso 4 hasta que se cumpla un criterio de paro.

La anterior definición deja abiertas diversas posibilidades para llevar a cabo las operacio-

nes del algoritmo. Se han probado distintas variantes y se ha visto que aún variaciones pequeñas en la forma de ejecutar estos pasos resultan en diferencias importantes en su comportamiento. Kuri [21, 22] hace un estudio comparativo de algoritmos genéticos estructuralmente distintos para analizar los efectos que tienen estas variaciones en el comportamiento del algoritmo. Concluye que del conjunto de algoritmos analizados, el algoritmo genético ecléctico (EGA) es aquél que presenta mejor desempeño (en precisión de la solución) contra un gran número de algunos problemas de optimización.

3.1.3. Teoría de convergencia

Si bien ha habido esfuerzos por construir una teoría de convergencia alrededor de los algoritmos genéticos, no la poseen en el sentido en el que los métodos exactos o los algoritmos de aproximación la tienen. Esto se debe a que los algoritmos genéticos son estocásticos. En la teoría de optimización numérica se habla de convergencia de un algoritmo al óptimo global cuando dicho algoritmo genera una sucesión de elementos en el espacio de soluciones cuyo valor límite es la solución de (3.1). Los algoritmos genéticos son estocásticos, esto es, poseen una componente de aleatoriedad en cada iteración. Por ello, la anterior definición de convergencia resulta inapropiada para analizarlos. Sin embargo, se puede utilizar una versión probabilística de ella, modelando el algoritmo genético como una cadena de Markov. Eiben[23], Davis [24] y Rudolph [25] han mostrado cómo se puede pensar a un algoritmo genético como una cadena de Markov. En particular, Rudolph presenta un análisis de convergencia para el CGA y muestra en qué casos es que un algoritmo genético converge al óptimo global y en qué casos no se puede garantizar su convergencia. A continuación se esboza la prueba de Rudolph, pues el detalle de la prueba se escapa de los límites de este trabajo.

Definición 3.2. Una **cadena de Markov** es un proceso estocástico a tiempo discreto que satisface la propiedad de Markov: $P(X_{n+1}|X_0, X_1, ..., X_n) = P(X_{n+1}|X_n)$, esto es, que la probabilidad de transición de un estado a otro depende únicamente del estado anterior.

Si bien las cadenas de Markov pueden ser discretas o continuas dependiendo de la cardinalidad del espacio de estados, en este trabajo interesan las cadenas de Markov discretas y en particular finitas, pues la cardinalidad del espacio de soluciones está acotado.

Para que una cadena de Markov quede completamente definida, es necesario especificar el espacio de estados $\mathcal{S}$, la matriz de probabilidades de transición P, y la distribución de probabilidad del estado inicial, normalmente denotada $p_0 = P(X_0 = x)$. Para modelar el algoritmo genético como una cadena de Markov, se analiza lo siguiente:

- El estado del algoritmo depende únicamente de la colección de códigos de todos los individuos de la población. Si se tienen n individuos en la población y la suma de las longitudes de todos los genes de cada individuo es l, entonces el espacio de estados es $\mathcal{S} = \mathbb{B}^{nl}$. Sin pérdida de generalidad, se puede suponer $\mathbb{B} = \{0, 1\}$ por ser la codificación binaria la que normalmente se utiliza, por ser la más fina. Así, cada elemento de $\mathcal{S}$ se puede pensar como un entero en representación binaria. Se define la proyección $\pi_k(i)$ como la k-ésima secuencia de l bits del estado i para poder hacer referencia al k-ésimo individuo en el estado i.
- Se pueden capturar las probabilidades de transición en una matriz P que admite una descomposicón natural en un producto de matrices estocásticas $P = CMS$ en donde C, M y S representan los operadores de cruza, mutación y selección respectivamente. El detalle de la construcción de estar matrices se puede consultar en el artículo original de Rudolph [25].
- La distribución inicial de la cadena de Markov, p_0 es irrelevante para analizar su comportamiento limítrofe.

Ahora sí, se puede definir la convergencia de un algoritmo genético como sigue:

Sea $t \in \mathbb{N}$ una variable que modela el paso del tiempo en el algoritmo genético; esto es, representa el índice de la generación de individuos sobre la que se está operando.

Definición 3.3. Sea $Z_t = \text{mín}\{f(\pi_k^{(t)}(i))|k = 1, ..., n\}$ una sucesión de variables aleatorias que representan el mejor valor de la función de adecuación (función objetivo) dentro de una población representada por el estado i en la generación t. Un algoritmo genético **converge** al mínimo global si y sólo si

$$\lim_{t\to\infty} P(Z_t = f^*) = 1$$

donde $f^* = \text{mín}\{f(x)|x \in \mathcal{D}\}$ es la solución global del problema (3.1).

Rudolph [25] concluye que el algoritmo genético canónico que expone Holland en [20], con probabilidad 1 no converge a la solución del problema. Sin embargo, cuando se

introduce una variante, se puede probar la convergencia en el sentido de la definición 3.3. Dicha variante es simplemente que el mejor individuo de la población en la generación t permanece en la población en la generación $t+1$. Esto garantiza que la calidad de las soluciones que encuentra el algoritmo al menos se mantenga constante de una generación a otra. El análisis de cadenas de Markov se complica un poco al añadir esta variante pero permite llegar al resultado de convergencia global para los algoritmos genéticos que preservan la mejor solución de cada generación en la siguiente. El detalle del final de la prueba se puede consultar en el artículo de Rudolph [25].

Dentro de este tipo de algoritmos cae el EGA que se utiliza como componente principal del CENG. Además, el EGA resultó ser el que convergía con mayor precisión a la solución en el análisis comparativo de Kuri [21], como se explicó en la sección anterior.

3.1.4. Optimización multiobjetivo

En la práctica, los algoritmos genéticos han probado ser muy buenas herramientas para resolver problemas de optimización multiobjetivo. Un problema de optimización multiobjetivo es aquel en el que se busca optimizar más de una función objetivo. Dado que los objetivos de las múltiples funciones a optimizar pueden entrar en conflicto, no existe una única solución para este tipo de problemas. En general, un problema de este tipo se puede plantear de la siguiente forma:

$$\begin{aligned} \text{mín } & F(x) = [f_1(x), f_2(x), ..., f_m(x)] \\ \text{s.a. } & x \in \mathcal{D} \end{aligned} \tag{3.2}$$

Nótese que a diferencia del problema 3.1, lo que se pretende minimizar aquí es un vector de funciones objetivo $f_i : \mathbb{R}^n \to \mathbb{R}$ con $i \in \{1, 2, ..., m\}$. Claramente, el concepto escalar de optimalidad (mínimo / máximo) no aplica para un problema de este tipo. En general, existen dos tipos de enfoques para resolver problemas como 3.2.

1. Combinar los diferentes objetivos en una función escalar Φ, por ejemplo $\Phi(x) = \sum_{i=1}^{m} \gamma_i f_i(x)$, con $\gamma_i \in \mathbb{R}$, y minimizar Φ. El problema principal es definir los escalares γ_i, pues con ellos se define el compromiso entre los diferentes objetivos del problema.

2. Optimizar las funciones simultáneamente para obtener un conjunto de puntos

que conforman el frente de Pareto (ver definiciones 3.4, 3.5, 3.6). Del conjunto de puntos solución, un experto en el dominio del problema elegiría una solución adecuada. Sin embargo, este enfoque suele ser ineficiente computacionalmente para $m > 3$.

Definición 3.4. Un punto x^* se dice que es un **óptimo débil de Pareto** para el problema multiobjetivo 3.2 si y sólo si $\nexists x \in \mathcal{D}$ tal que $f_i(x) \leq f_i(x^*) \forall i \in \{1, 2, ..., m\}$

Definición 3.5. Un punto x^* se dice que es un **óptimo estricto de Pareto** para el problema multiobjetivo 3.2 si y sólo si $\nexists x \in \mathcal{D}$ tal que $f_i(x) < f_i(x^*) \forall i \in \{1, 2, ..., m\}$

Definición 3.6. Se le llama **frente de Pareto** al conjunto de todos los óptimos de Pareto, tanto estrictos como débiles.

Una alternativa que ha probado funcionar en la práctica, es utilizar algoritmos genéticos con un enfoque mín − máx o minimización en norma L_∞. Esto es, el problema de optimización multiobjetivo se podría resolver como

$$\begin{aligned} &\text{mín } ||[f_1(x), f_2(x), ..., f_m(x)]||_\infty \\ &\text{s.a. } x \in \mathcal{D} \end{aligned} \tag{3.3}$$

o equivalentemente

$$\begin{aligned} &\text{mín } \max\{f_i(x) | i \in \{1, 2, ..., m\}\} \\ &\text{s.a. } x \in \mathcal{D} \end{aligned} \tag{3.4}$$

Ignacio López Peña, Maestro en Ciencias por la UNAM, desarrolló en conjunto con el Dr. Kuri un estudio (sección 6.3 de [26]) en el que mostró que la solución resultante cae sobre el frente de Pareto, para el conjunto de problemas multiobjetivo definidos por Zitzler et.al. en [27]. Este enfoque claramente no permite calcular todo el frente, sin embargo la solución que arroja resulta ser un óptimo de Pareto en la gran mayoría de los casos.

El CENG utiliza el EGA con un criterio mín − máx para encontrar el conjunto de códigos que mejor preservan los patrones presentes en una base de datos. Más adelante se verá cómo al plantear el problema 4.3, se está optando por resolver un problema de optimización multiobjetivo utilizando justamente este enfoque.

3.2. Aprendizaje supervisado

El aprendizaje supervisado resulta un concepto mucho más intuitivo que el no supervisado, pues se cuenta con un "maestro" que enseña al sistema. En la figura 3.1 se puede ver un diagrama de un sistema de aprendizaje supervisado. En ese modelo, si se agrupan las variables predictoras en un vector $x = (x_1, x_2, ..., x_n)$ y se conoce la respuesta esperada y, se puede pensar que el sistema de aprendizaje recibe el vector x y arroja una salida $\hat{y}$. El rol del maestro es retroalimentar al sistema de aprendizaje con una medida de error entre la respuesta esperada y y la respuesta del sistema $\hat{y}$. Con base en el error $||y - \hat{y}||$ medido a la salida del sistema, se puede aplciar una regla de corrección para mejorar su precisión.

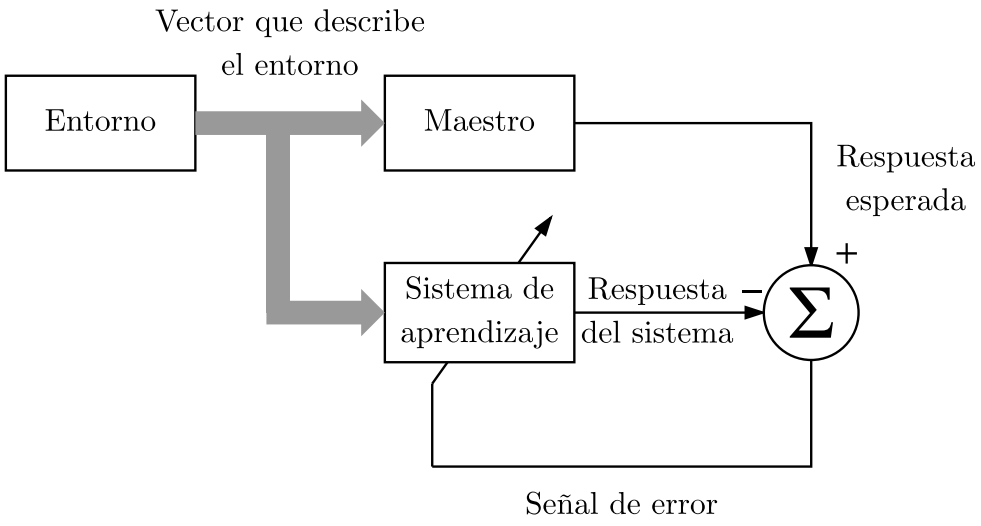


Figura 3.1. Modelo de aprendizaje supervisado [2]

Por lo general, se suele dividir en dos grandes grupos a los métodos de aprendizaje supervisado. Se habla de **regresión** cuando la variable Y a predecir es una variable cuantitativa. Cuando la variable Y es cualitativa, el problema se suele llamar de **clasificación**.

Algunos ejemplos de algoritmos de aprendizaje supervisado son:

- Redes neuronales
- Árboles de decisión

- Máquinas de soporte vectorial
- Clasificador bayesiano

En el libro de Hastie, et.al. [28] se pueden consultar estos métodos a detalle.

3.3. Redes de perceptrones multicapa como aproximadores de funciones

Para poder identificar y preservar los patrones presentes en los datos, el CENG se apoya en redes neuronales, en particular en redes de perceptrones multicapa. Como se menciona en la sección 3.2, son una herramienta muy útil para realizar aprendizaje supervisado, pues son especialmente buenas para detectar patrones presentes en un conjunto de datos. Asimismo, la expresión matemática de la salida de la red resulta especialmente útil para aproximar funciones multivariadas complicadas como se verá en la sección 3.3.2. Es por ello que se emplean en el CENG.

3.3.1. Características generales

Las redes de perceptrones multicapa son un tipo particular de redes neuronales compuestas de una capa de entrada, una o varias capas ocultas de nodos de cómputo llamados perceptrones y una capa de perceptrones de salida. En la figura 3.2 se puede ver un esbozo de la arquitectura de una red de este tipo.

De acuerdo a Haykin [2], poseen tres características fundamentales:

1. El modelo de cada neurona (perceptrón) se basa en una función de activación suave y no-lineal. Usualmente se utiliza la función logística $y = \frac{1}{1+e^{-x}}$ en donde y representa el valor de salida (axón) de la neurona y x es la suma ponderada de todas las entradas (dendritas). La no-linealidad de la función de activación es importante, pues de otro modo, se puede reducir el modelo al de un perceptrón de una sola capa.
2. Contiene una o más capas de neuronas ocultas que no forman parte de las entradas o salidas de la red. Dichas capas son las que permiten que la red extraiga patrones complejos detectando características más significativas a partir de las encontradas

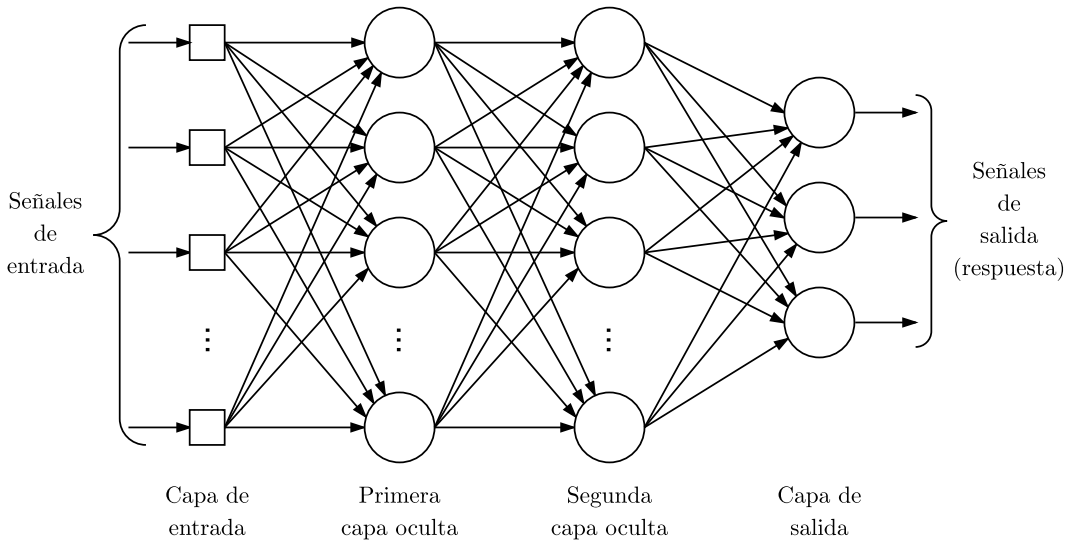


Figura 3.2. Arqutiectura básica de una red de perceptrones multicapa[2]

por la capa anterior.

3. Tiene un alto grado de conectividad. Esto es, existen suficientes conexiones sinápticas entre las neuronas que conforman la red.

Recobraron su popularidad en la década de los 80, tras el descubrimiento por Rumelhart, Hinton y Williams del algoritmo de retropropagación para entrenarlas [29]. Dicho algoritmo consta de dos fases. En la primera fase se alimenta un vector de entrenamiento a la red y su efecto se propaga hacia adelante por la red hasta producir una salida. Dicha salida se mide contra la salida esperada de la red. En la segunda fase se recorre la red hacia atrás ajustando los pesos sinápticos de todas las conexiones de acuerdo a una regla de corrección de errores basada en el error medido al final de la primera fase. En la figura 3.3 se puede ver el flujo de información durante las dos fases del algoritmo de entrenamiento. El detalle del algoritmo de retropropagación, se puede consultar en el trabajo original de Rumelhart, Hinton y Williams [29].

Justamente la combinación de la arquitectura de la red con la eficiencia computacional del algoritmo de retropropagación es la que hace atractivo a este método.

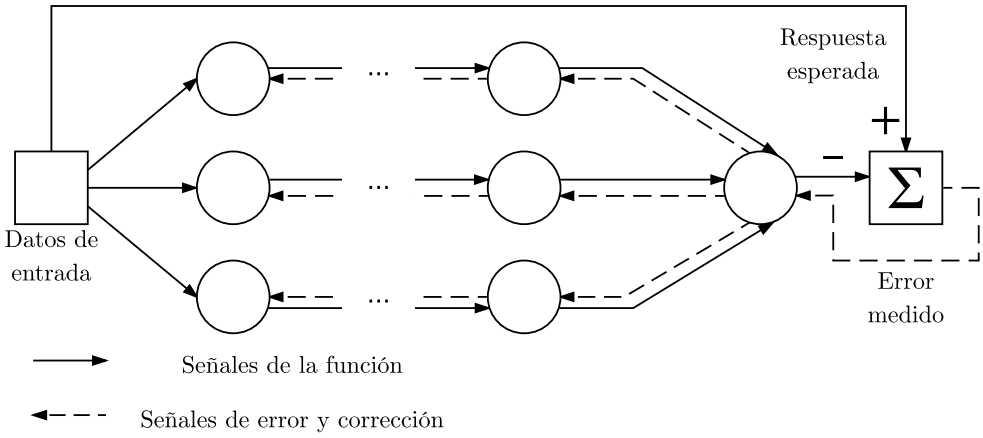


Figura 3.3. Flujo de información durante el entrenamiento de la red[2]

3.3.2. Teoría de aproximación

En 1989, Cybenko prueba una de las primeras versiones del teorema universal de aproximación [30] para un conjunto de funciones de activación sigmoidales entre otras familias de funciones. Posteriormente Hornik[31, 32] muestra que es más una propiedad de la arquitectura de las redes de perceptrones multicapa que de la función de activación utilizada. La versión que presenta Haykin [2] del teorema es la siguiente:

Teorema 1. *Sea ϕ una función continua, monótona creciente y acotada. Sea I_m el hipercubo unitario de dimensión m, $[0,1]^m$. Se denota el espacio de funciones continuas en I_m como $C(I_m)$. Entonces, $\forall f \in C(I_m)$, $\forall \varepsilon > 0$, $\exists\ N \in \mathbb{N}, v_i, b_i \in \mathbb{R}$ y $w_i \in \mathbb{R}^m\ \forall i \in \{1, ..., N\}$ tales que se puede definir*

$$F(x) = \sum_{i=1}^{N} v_i \phi(w_i^{\intercal} x + b_i) \tag{3.5}$$

como una aproximación de f, esto es

$$|F(x) - f(x)| < \varepsilon\ \forall x \in \mathbb{R}^m$$

Note que la función logística $\phi(x) = \dfrac{1}{1+e^{-x}}$ que se utiliza comúnmente como función de activación en este tipo de redes cumple con las condiciones del teorema. Más aún, la ecuación 3.5 representa la salida de una red de perceptrones como la que se muestra en la figura 3.4. Nótese que en la red de la figura la salida presenta un término independiente

b_0 que Haykin omite en el teorema 1. Dicho término no aparece siempre en la literatura pero es útil pues captura un sesgo independiente a las variables de entrada que puede tener el modelo. Una red como la que se presenta en la figura 3.4 tiene las siguientes características:

- La red cuenta con m nodos de entrada y una única capa oculta con N neuronas.
- Las neuronas de la capa oculta tienen pesos $w_i \in \mathbb{R}^m$ y sesgo b_i para $i \in \{1, ..., N\}$
- La salida de la red neuronal es una combinación lineal de las salidas de las neuronas ocultas, donde el vector $v \in \mathbb{R}^N$ define los pesos sinápticos de la capa de salida; i.e. los coeficientes de la combinación lineal.

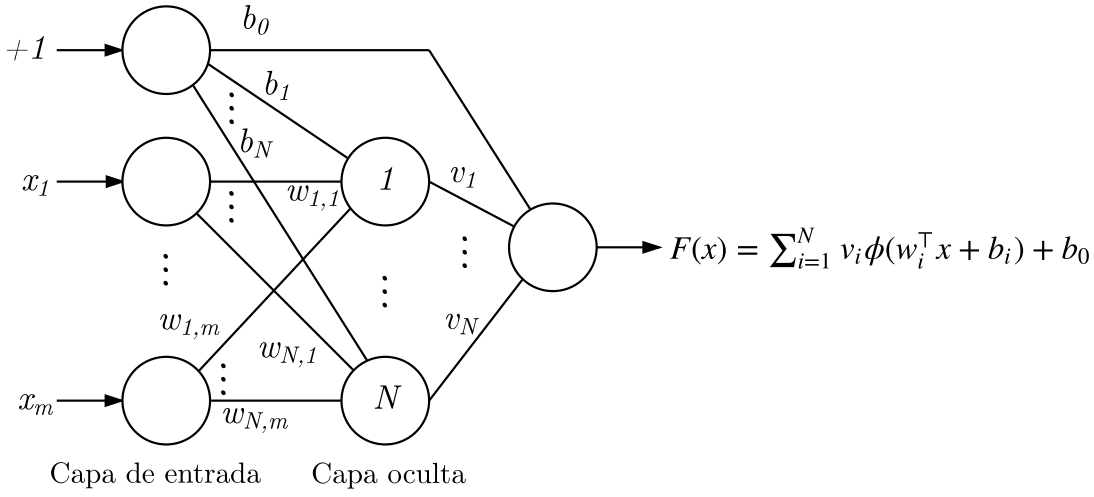


Figura 3.4. Red de perceptrones de una sola capa oculta [3]

Si bien el teorema 1 garantiza la existencia de una arquitectura que aproxime a cualquier función continua en I_m, no es constructivo; i.e. no expone la manera en la que se debe construir dicha red.

Para poder aproximar un conjunto de datos utilizando el teorema universal de aproximación, hay que garantizar lo siguiente:

1. Los datos a aproximar se pueden representar con una función continua.
2. Estimar el tamaño efectivo de los datos (cuánta información no redundante hay realmente en los datos), para no sobreestimar el número de neuronas ocultas.
3. Determinar el número de neuronas en la capa oculta.
4. Mapear los datos de entrada al intervalo $[0, 1]$.

5. Utilizar un algoritmo de entrenamiento efectivo y eficiente para la red neuronal.

Kuri [3] expone una forma de cumplir con las anteriores condiciones de la siguiente manera:

1. Para garantizar la continuidad de la función a modelar, se emplea el spline natural de los datos originales. Si se tienen $n+1$ puntos a interpolar $\{(x_1, y_1), ..., (x_{n+1}, y_{n+1})\}$, el spline natural es una colección de polinomios de grado 3 definida por partes de la siguiente manera:

   $$S(x) = S_i(x) \ \forall x \in [x_i, x_{i+1}] \ \forall i \in \{1, ..., n\}$$

   en donde cada S_i es un polinomio de grado 3 que además debe cumplir las siguientes características:

   - $S_i(x_i) = y_i$ y $S_i(x_{i+1}) = y_{i+1} \forall i \in \{1, ..., n\}$ i.e. que el spline interpole todos los puntos del conjunto de datos.
   - $S_i'(x_{i+1}) = S_{i+1}'(x_{i+1}) \forall i \in \{1, ..., n-1\}$ i.e. que la función resultante sea suave y derivable en todos sus puntos.
   - $S_i''(x_{i+1}) = S_{i+1}''(x_{i+1}) \forall i \in \{1, ..., n-1\}$ i.e. que la concavidad del spline no cambie abruptamente.
   - Que la curvatura de $S(x)$ sea la mínima entre todas las posibles funciones de colocación.

   Se puede definir un sistema de ecuaciones que permite obtener los coeficientes de los polinomios que conforman $S(x)$. Para que dicho sistema de ecuaciones tenga solución única se deben agregar dos condiciones adicionales, las cuales para el caso del spline natural (aquél de mínima curvatura) corresponden a $S_1''(x_1) = 0$ y $S_n''(x_{n+1}) = 0$. Existe además un método para evaluar $S(x)$ en cualquier punto sin resolver el sistema de ecuaciones anteriormente mencionado, haciendo que el cómputo de la interpolación sea eficiente.

   Se introducen, entonces, además de los datos originales, puntos equidistantes que se encuentran sobre el spline para garantizar la continuidad de los datos.

2. La cantidad real de información que contiene un conjunto de datos se puede expresar con la complejidad de Kolmogorov, que mide el tamaño del programa mínimo que puede generar ese conjunto de datos [33]. Resulta imposible calcular la complejidad de Kolmogorov, pues no es posible encontrar el programa mínimo

que genera cierta salida pues en caso de existir, esto implicaría que se ha resuelto el problema de la detención (mejor conocido como "Halting Problem"), como se puede ver en el artículo de Chaitin [34]. Turing [35] probó en 1936 que el problema de la detención es indecidible; esto es, no existe una máquina de Turing que lo pueda resolver. Por ello, se emplea la mejor aproximación práctica que existe. Para ello se utiliza el algoritmo Prediction by Partial Matching (PPM), el cual es el estado del arte en compresión de datos sin pérdida [36, 37]. El factor de compresión de este algoritmo se utiliza justamente para aproximar la complejidad de Kolmogorov.

3. En [3], Kuri presenta una fórmula que permite estimar un valor mínimo para el número de neuronas en la capa oculta. Dicha fórmula es una estimación estadística con base en el número de variables de entrada a la red y el número de parejas de datos de entrenamiento con los que se cuenta.

4. El mapeo de los datos al intervalo $[0, 1]$ se hace de manera lineal, mapeando al 0 el mínimo valor que toma cierta variable numérica y al 1 el máximo.

5. Recientemente ha recobrado popularidad el algoritmo de retropropagación para entrenar redes neuronales [29]. Su eficiencia y efectividad son adecuadas para entrenar las redes con las que se estará trabajando.

3.4. Aprendizaje no supervisado

Cuando no se cuenta con datos etiquetados con la respuesta esperada del sistema, se buscan patrones y estructuras interesantes en los datos a través de métodos de aprendizaje no supervisado. Si bien puede parecer como una tarea más compleja que el aprendizaje supervisado, el modelo de aprendizaje es muy sencillo y se muestra en la figura 3.5.

Algunos ejemplos de métodos de aprendizaje no supervisado son:

- Agrupamiento
- Expectation-Maximization
- Método de momentos
- Análisis de componentes principales

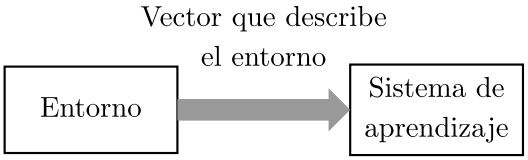


Figura 3.5. Modelo de aprendizaje no supervisado [2]

- Descomposición en valores singulares
- Mapas auto-organizados
- Redes difusas

Del mismo modo que con los métodos de aprendizaje supervisado, estos métodos se pueden consultar a detalle en el libro de Hastie et.al. [28].

3.4.1. Agrupamiento

El agrupamiento o análisis de grupos es un método de aprendizaje estadístico no supervisado. Esto es, el algoritmo aprende a partir de datos no etiquetados, infiriendo los patrones (propiedades de la distribución de probabilidad) presentes en los datos. Difiere del aprendizaje supervisado en que el objetivo no es predecir el valor de unas variables en términos de otras. Por ello, se puede prescindir de etiquetas en los datos de entrenamiento que permitan estimar un error de predicción. Sus objetivos son variados pero todos tienen que ver con agrupar los elementos de un conjunto de datos en cúmulos o 'clusters' de tal forma que todos los elementos pertenecientes al mismo cúmulo estén relacionados de forma más cercana entre ellos que con elementos de cúmulos distintos. El punto fundamental para cualquier análisis de grupos es la métrica de similitud entre los elementos del conjunto a agrupar.

El agrupamiento resulta un concepto intuitivo cuando se trabaja únicamente con variables numéricas, pues estamos acostumbrados a trabajar con métricas como la distancia euclidiana. ¿Qué sucede cuando se quiere trabajar con conjuntos de datos que contienen variables tanto categóricas como numéricas? En el pasado este problema se ha tratado de dos maneras distintas: intentando el análisis directamente (agrupamiento no métrico) y codificando las variables categóricas de alguna manera tal que resulte una base de

datos puramente numérica a la que se le pueda aplicar un algoritmo de agrupamiento numérico.

I. Agrupamiento no métrico

Cuando no se desean transformar los datos para llevar a cabo un análisis de grupos, se cuenta con propuestas intuitivas para agrupar los datos. Una limitante importante de estos tratamientos es que está basado en consideraciones *a priori* sobre los datos que pueden sesgar y limitar las conclusiones del análisis. Una alternativa es el enfoque de Ienco [38], quien propone un método automático basado en teoría de información para encontrar un contexto apropiado que le permita definir una métrica entre atributos categóricos. Sin embargo, no se profundizará en este tipo de agrupamiento, pues no es de interés para este trabajo.

II. Agrupamiento métrico

El agrupamiento métrico es aplicable en general, pero requiere de la codificación de las variables categóricas en variables numéricas. En ocasiones se define *a priori* con una matriz de similitud el grado de diferencia entre los pares de valores que puede tomar cada variable categórica. Este enfoque, sin embargo, se basa nuevamente en consideraciones *a priori* sobre los datos. Existen diversos sistemas de codificación que se han utilizado en el pasado. CENG[19] propone un algoritmo de codificación de variables categóricas que pretende superar diversas limitantes de los esquemas de codificación tradicionales.

Una vez que se cuenta con variables codificadas numéricamente, se les puede aplicar un algoritmo de agrupamiento estrictamente numérico. Para ello existen varios tipos de algoritmos; por ejemplo: jerárquicos, basados en centroides (k-means), basados en distribuciones (Expectation-Maximization), basados en densidad (Mean Shift). Otro punto fundamental a considerar es la métrica de similitud que se utilizará. Resulta muy intuitivo utilizar la distancia euclidiana como métrica de similitud cuando se trabaja con variables numéricas. Sin embargo, no siempre puede resultar apropiada para encontrar ciertos tipos de patrones presentes en los datos, especialmente cuando tienen distribuciones complejas.

3.4.2. Agrupamiento entrópico

Un algoritmo de agrupamiento que ha mostrado tener muy buen desempeño para encontrar cúmulos con distribuciones complicadas es el agrupamiento basado en entropía que propone Aldana [39]. Se basa en una métrica utilizada comúnmente en teoría de la información: la entropía. Dicha métrica se suele usar como una medida del desorden de un conjunto de datos. Por ello, un enfoque intuitivo cuando se utiliza en algoritmos de agrupamiento es minimizarla dentro de cada uno de los cúmulos. Esto permite definir un cúmulo como un conjunto de datos que tienen el mínimo desorden posible.

Los algoritmos de agrupamiento tradicionales buscan los cúmulos que optimizan cierta métrica y posteriormente evalúan la calidad del agrupamiento con un índice de validez. El agrupamiento entrópico, en cambio, busca los cúmulos que optimizan ese índice de validez. En lugar de utilizar la entropía como la métrica a optimizar, el algoritmo de Aldana la utiliza como parte del índice de validez. Para ello, se explora el espacio de todas las posibles distribuciones de probabilidad para un agrupamiento de los datos y se elige aquella que maximiza la entropía total de la distribución. Si se interpreta la entropía como el grado de desorden (i.e. incertidumbre) de un conjunto de datos, resulta contraintuitivo querer maximizar esta cantidad. Sin embargo, esta decisión está basada en el *principio de máxima entropía* expuesto por Jaynes [40]. Éste dice que si se va a hacer inferencia por no tener la información completa, la única forma de no sesgarla es eligiendo la distribución de probabilidad de máxima entropía, sujeta a las restricciones que se tienen sobre la distribución. Así, se elige una distribución de probabilidad que evita el sesgo por consideraciones subjetivas al analizar todas las posibilidades, pues al buscar aquella de entropía máxima sujeto a toda la información que sí se conoce se está eligiendo aquella con el mayor grado de incertidumbre. Elegir cualquier otra distribución trae implícitas suposiciones basadas en información con la que, por hipótesis, no se cuenta.

Formalmente, este algoritmo se puede plantear de la siguiente forma:

Definición 3.7. La **distribución de probabilidad** Π es una posible asignación de puntos a cúmulos encontrado. Esto es,

$$\Pi = \{C_i \mid i \in \{1, 2, ..., k\}\}, C_i = \{x \mid x \text{ pertenece al cúmulo } i\}$$

en donde k es el número de cúmulos a encontrar.

Sea S es espacio de todas las posibles asignaciones de puntos a un cúmulo. Entonces

$$S = \{\Pi \mid \Pi \text{ es una distribución de probabilidad en el sentido de la defínción 3.7}\}$$

Definición 3.8. La **entropía de una distribución de probabilidad** Π es

$$H(\Pi) = \sum_{C_i \in \Pi} H(X|C = C_i) = -\sum_{C_i \in \Pi} \sum_{x \in C_i} p(x|C_i) \log(p(x|C_i))$$

Definición 3.9. Sea $\overrightarrow{\sigma}$ un vector de la forma $\overrightarrow{\sigma} = (\sigma_1, \sigma_2, ..., \sigma_n)$ con $\sigma_j =$ la desviación estándar de los objetos en un cúmulo en la dimensión j. Se define la **suma de las desviaciones estándar de los cúmulos** para una distribución Π como

$$g(\Pi) = \sum_{i=1}^{k} \overrightarrow{\sigma}(C_i) = \sum_{i=1}^{k} \sum_{j=1}^{n} \sigma_j \ \forall \sigma_j \in \overrightarrow{\sigma}$$

De este modo, el agrupamiento entrópico busca la mejor distribución de agrupamiento optimizando el problema multiobjetivo:

$$\begin{aligned} &\text{máx } H(\Pi) \wedge \text{mín } g(\Pi) \\ &\text{s.a. } \Pi \in S \end{aligned} \tag{3.6}$$

Al plantear el problema de este modo, se van eligiendo distribuciones de agrupamiento que minimizan la suma de desviaciones estándar intra-cúmulo, al mismo tiempo que se elige una distribución sólo tomando en cuenta la información disponible en ese momento. Aldana se basa en el concepto de dominancia que utilizan la mayoría de los algoritmos de optimización multiobjetivo al buscar un óptimo de Pareto para el problema 3.6. Para explorar el espacio de soluciones, utiliza el algoritmo genético ecléctico (EGA) que se menciona en la sección 3.1.2. El detalle del funcionamiento de este algoritmo de agrupamiento se puede consultar en el artículo original de Aldana et.al. [39].

3.4.3. Mapas auto-organizados

Los mapas auto-organizados (también conocidos como mapas de Kohonen) son una herramienta de aprendizaje no supervisado para reducir la dimensionalidad de un conjunto de datos preservando las características estructurales del espacio original. Estos

mapas típicamente se usan para resolver problemas de agrupamiento, pues a los nodos que componen el mapa se les pueden asignar etiquetas de clase. Su modelo de aprendizaje por refuerzo y competencia resulta más apropiado que el de otros algoritmos de agrupamiento como $k-$medias, pues permite detectar cúmulos con formas más complejas. La representación en un menor número de dimensiones (usualmente dos) permite visualizar los resultados más claramente que cuando se trabaja en espacios de alta dimensión. En este sentido, los mapas auto-organizados se asemejan al análisis de componentes principales.

Su funcionamiento es sencillo y se explica en el artículo de Kohonen [41]. Se define una retícula de m nodos (o neuronas) conectadas en (usualmente) dos dimensiones. Dicha retícula usualmente es rectangular o hexagonal. Cada nodo N_i tiene un vector de pesos w_i de la misma dimensión n que los vectores de entrada v_i.

El algoritmo de entrenamiento del mapa auto-organizado es el siguiente:

1. Se inicializan los pesos de alguna manera (se suele hacer de forma aleatoria).
2. Para cada vector del conjunto de datos, se encuentra su nodo más cercano, midiendo la distancia entre el vector y el vector de pesos del nodo: $||v_i - v_j||$. A este nodo se le llama BMU por sus siglas en inglés (Best Matching Unit).
3. Se actualizan los pesos de los nodos cercanos al BMU de acuerdo a la siguiente regla:

   $$w_j(t+1) = w_j(t) + \Theta(j)\alpha(t)(v_i - w_j(t))$$

   Donde $\alpha(t)$ es una función monótona decreciente que indica la tasa de aprendizaje y $\Theta(j)$ es una función monótona decreciente para la distancia entre el BMU y el nodo j de acuerdo a la topología de la retícula. Esto es, la tasa de corrección en los pesos va a ser menor mientras pase el tiempo y mientras más lejos esté el nodo j del BMU.
4. Se incrementa al valor de t en uno y se repite el proceso de aprendizaje (paso 2) hasta que el valor de t hace que la vecindad de la BMU que se actualiza contiene únicamente a la BMU.

Para las funciones Θ y α se utiliza comúnmente la función de decaimiento exponencial:

$$\alpha(t) = \alpha_0 e^{-t/\lambda_\alpha}$$

$$\Theta(j) = \theta_0 e^{-d(j)/\lambda_\theta}$$

donde $d(j) = \text{dist}(BMU, N_j)$ es la distancia del nodo j a la BMU.

Al final del entrenamiento se cuenta con una retícula en la que cada nodo tiene un peso que ya ha sido ajustado por el entrenamiento. Si se contara con datos etiquetados con el cúmulo al que pertenecen, se podría encontrar el BMU para cada punto y asociarle al BMU el mismo cúmulo que el que tiene el punto. Así, resultaría una visualización en dos dimensiones de los cúmulos de un espacio de dimensión más elevada.

Para este algoritmo, usualmente se mide la distancia entre puntos y nodos utilizando la distancia euclidiana ($||\ \cdot\ ||_2$). Sin embargo, podrían utilizarse otras métricas que sean más apropiadas para el problema que se esté intentando resolver.

4. Diseño de la Solución

Design is not just what it looks like and feels like. Design is how it works.

– Steve Jobs

En este capítulo se explican los diferentes componentes que integran la solución y la manera en la que interactúan. Asimismo, se describe la forma en la que fluye y se procesa la información en la solución propuesta.

4.1. Flujo básico

El objetivo central de este trabajo es poder identificar al autor de un texto desconocido utilizando métodos numéricos. Para poder hacer esto se requiere que el texto de entrada (información de carácter no estructurado por naturaleza) adquiera no sólo estructura, sino también una representación numérica.

A través de una serie de consideraciones *a priori* y decisiones de diseño basadas en mediciones estadísticas, se puede muestrear el texto original y encajonarlo en una estructura rígida la cual se espera que preserve los patrones que el autor le imprimió al escribirlo.

Del paso anterior, resulta una representación simplificada del texto como una base de datos estructurada, en la que existen variables tanto numéricas como categóricas. Ésta se le alimenta al CENG para obtener la codificación numérica que mejor preserva los patrones que existen entre las variables presentes en ella.

Por último, la base de datos numérica resultante es procesada por algoritmos ya conocidos para identificar los patrones ahí presentes y obtener una representación abstracta del estilo de escritura del autor. Con base en esa representación abstracta se define una forma de medir similitud entre dos textos para poder identificar si una pareja de textos

pertenecen o no al mismo autor.

En la figura 4.1, se pueden ver los tres componentes antes descritos y la manera secuencial en la que están conectados.

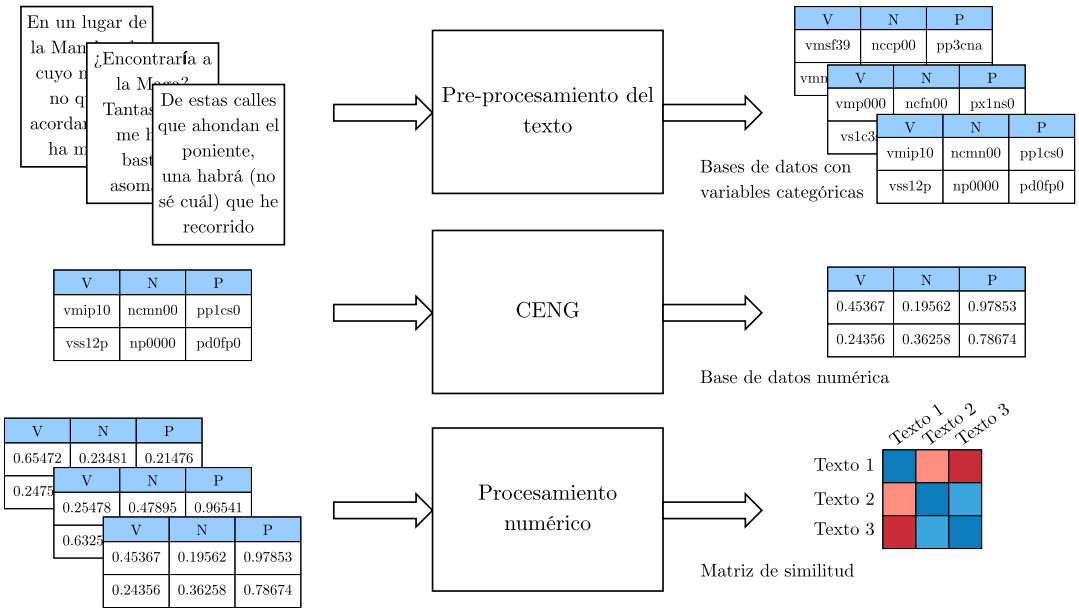


Figura 4.1. Flujo básico de información en el sistema propuesto

4.2. Corpus de textos a analizar

Para este trabajo, se recopiló un corpus original de textos en español, pues no se encontró ninguno existente que tuviera las característica deseadas. Dicho corpus consiste de 234, 049 palabras de 10 autores hispanoparlantes. En la tabla 4.1 se puede ver el desglose de la cantidad de palabras por autor. El corpus consta de estilos de escritura muy distintos que incluyen el clásico estilo de Cervantes, el de los autores del *Boom Latinoamericano*, la poesía de *Los Contemporáneos*, el estilo de un ensayista y poeta amateur y el de un periodista en formación, entre otros.

Para cada uno de los autores hay textos de diferentes longitudes. De este modo se pueden encontrar textos de longitudes similares para los distintos autores, de tal modo que la comparativa entre uno y otro sea lo más precisa posible.

Sobre el corpus completo se realizaron estudios estadísticos descritos en las secciones 4.3 y 4.5.

4.3. Preprocesamiento del texto

Tras analizar las características lingüísticas que han sido utilizadas con éxito en el pasado para determinar la huella estilística de un autor (sección 2.3), se optó por centrar el diseño de la estructura del texto simplificado en las categorías gramaticales de las palabras que se utilizan en el texto original. Del mismo modo, se decidió hacer el análisis del texto a nivel de oración, pues se cree que esto provee una granularidad necesaria y suficiente. Además de las categorías gramaticales de las palabras presentes en una oración, se analizan otras características **léxicas** que han sido utilizadas en el pasado como número de signos de puntuación y longitud de oración.

Autor	**Número de palabras**
Gabriel García Márquez	43,366
Horacio Quiroga	18,008
Julio Cortázar	41,100
Jorge Luis Borges	16,920
Miguel de Cervantes	59,279
Octavio Paz	18,025
Mario de la Piedra	10,890
Luis Alberto Madrigal	12,317
Héctor Abad Faciolince	9,370
José Gorostiza	4,774
Total	234,049

Tabla 4.1. Desglose de autores incluidos en el corpus analizado

4.3.1. Stanford POS Tagger

Para identificar la categoría gramatical de cada palabra se pretendía utilizar el *Diccionario de la Real Academia de la Lengua Española*. Sin embargo, el problema que esto presenta es que existen palabras que, dependiendo del contexto, pueden adquirir diferente función gramatical. Tal es el caso de la palabra "rojo". Dependiendo del contexto, se puede estar calificando a un objeto como rojo, en cuyo caso la palabra funciona como adjetivo; o se puede estar hablando del color rojo, en cuyo caso la palabra funciona como sustantivo.

Por esta razón y el elevado tiempo de procesamiento requerido para consultar el diccionario en línea palabra por palabra, se decidió utilizar un etiquetador de categorías gramaticales desarrollado por el grupo de procesamiento de lenguaje natural de la Universidad de Stanford en California [42, 43]. Dicho etiquetador corre en Java, es ampliamente utilizado y se distribuye como parte de Stanford CoreNLP [44], y cuenta con un modelo pre-entrenado en español.

El modelo en español fue entrenado con el corpus AnCora [45]. Dicho corpus contiene alrededor de $500,000$ palabras y $17,000$ oraciones. Dado que las etiquetas utilizadas en este corpus están basadas en los estándares EAGLES definidas en [46], son extremadamente precisas y por ello el número de etiquetas distintas es muy elevado. Por esta razón, el equipo de NLP de Stanford tomó la decisión de reducir el tamaño del conjunto de etiquetas en los datos de entrenamiento a sólo 85, agrupando a las más similares dentro de la misma etiqueta.

En la tabla 4.2 se pueden ver las 12 categorías gramaticales definidas en los estándares EAGLES, así como la primera letra de las etiquetas que corresponden a cada categoría gramatical. Además, con base en el corpus de textos que se utilizará para este trabajo, se hizo un recuento del número etiquetas distintas encontradas por categoría después de haber etiquetado todos los textos. También se calculó el porcentaje que representan el número de etiquetas distintas por categoría del total de etiquetas encontradas (únicamente se encontraron 81 etiquetas distintas en la muestra de textos). Nótese que además se encontró una "categoría" adicional a las especificadas en los estándares EAGLES, identificada por un 3 como el primer caracter de la etiqueta. Sin embargo, como se ve en la tabla 4.3, dicha etiqueta sólo aparece 3 veces en todo el corpus y por ello fue ignorada.

El etiquetador funciona con cadenas de Markov y se basa en el principio de máxima entropía. Con ello, se toma en cuenta la secuencia de palabras que se han encontrado en el texto hasta el momento para determinar la categoría a la que pertenece cierta palabra. Ésto resulta muy efectivo para casos ambiguos como el ilustrado anteriormente al hablar del diccionario de la RAE. El detalle de su funcionamiento excede los límites de este trabajo y se puede consultar en [42, 43].

Categoría gramatical	**Primera letra de la etiqueta**	**Número de etiquetas distintas**	**Porcentaje del total de etiquetas**
Adjetivo	A	2	2.47 %
Conjunción	C	2	2.47 %
Determinante	D	7	8.64 %
Puntuación	F	15	18.52 %
Interjección	I	1	1.23 %
Sustantivo	N	5	6.17 %
Pronombre	P	9	11.11 %
Adverbio	R	2	2.47 %
Preposición	S	1	1.23 %
Verbo	V	33	40.74 %
Fechas	W	1	1.23 %
Números	Z	2	2.47 %
Desconocida	3	1	1.23 %
Total		81	100 %

Tabla 4.2. Categorías gramaticales presentes en los estándares EAGLES

El etiquetador toma como entrada un archivo que contiene el texto por analizar. A la salida, produce un archivo en el que a cada elemento del texto (palabra, signo de puntuación, número, entre otros) se le ha asignado la etiqueta que lo caracteriza. Así, el archivo de salida consta de una serie de elementos o **tokens** separados por espacios, donde cada token consiste de dos partes separadas por un guion bajo: el elemento del texto analizado y su correspondiente etiqueta. En la figura 4.2 se ilustra la diferencia entre el contenido del archivo de entrada y el de salida del etiquetador.

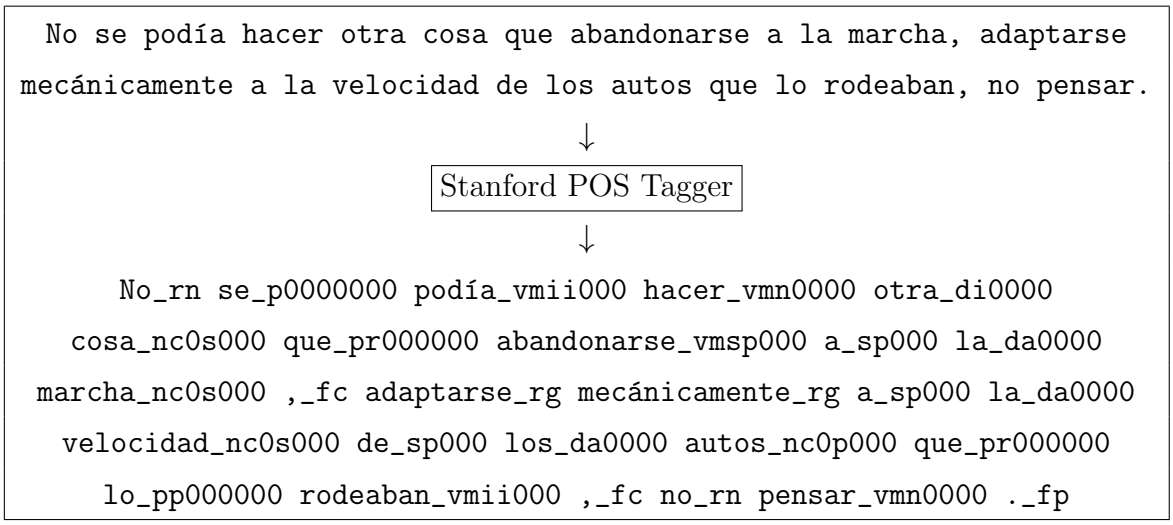


Figura 4.2. Entrada y salida del Stanford POS Tagger

4.3.2. Reducción del número de instancias categóricas

En los sucesivo, se utilizará el término **instancia categórica** para hacer referencia a uno de los posibles valores que puede tomar una variable categórica. En general, cuando se hable del número de instancias de una variable, se está hablando del número de valores *distintos* que puede tomar dicha variable. Análogamente, el número de instancias presentes en una base de datos significará la suma de todas las instancias de cada variable categórica.

A través de un recuento de las etiquetas presentes en el corpus de textos de entrada, se analizó la frecuencia con la que aparecen las distintas etiquetas para cada categoría gramatical. Los resultados se pueden ver en la tabla 4.3. Ahí se puede ver, por ejemplo, que en la muestra de textos el 21.6 % de los tokens analizados son sustantivos.

Tomando en cuenta los límites al tamaño del problema (número de columnas, renglones e instancias categóricas) que se puede resolver en un tiempo adecuado, se puede definir propiamente la estructura de la base de datos que modelará los textos. En la sección 4.5 se explica el detalle de la construcción de dicha estructura.

Categoría gramatical	**Primera letra de la etiqueta**	**Frecuencia de aparición**	**Porcentaje del total de tokens**
Adjetivo	A	16,915	6.75 %
Conjunción	C	16,588	6.62 %
Determinante	D	34,267	13.67 %
Puntuación	F	31,127	12.42 %
Interjección	I	221	0.09 %
Sustantivo	N	54,152	21.60 %
Pronombre	P	18,370	7.33 %
Adverbio	R	11,799	4.71 %
Preposición	S	32,065	12.79 %
Verbo	V	34,717	13.85 %
Fechas	W	206	0.08 %
Números	Z	236	0.09 %
Desconocida	3	3	0.00 %
total		250,656	100 %

Tabla 4.3. Número de instancias gramaticales presentes en el conjunto de textos de prueba

4.4. CENG

El algoritmo CENG asigna códigos numéricos a variables categóricas de tal forma que se obtenga una base de datos puramente numérica sin incrementar el número de dimensiones (columnas) de la base de datos original. El CENG está diseñado para preservar de la mejor manera posible los patrones presentes entre las variables de la base de datos. Se desarrolló originalmente en [19, 1, 47].

Es muy importante destacar que los códigos numéricos asociados a las variables categóricas que genera el CENG son válidos únicamente para la colección de datos que se le alimentó al algoritmo. De ninguna forma representan una manera de codificar apariciones sucesivas del mismo valor para cierta variable categórica en otra base de datos. Más aún, si a la colección de datos que se le alimentó al CENG originalmente, se le agregara o eliminara una observación, los códigos generados serían distintos. De este modo, la codificación resultante no tiene los problemas que presentan otros esquemas de codificación al estar basados en consideraciones *a priori*. Este esquema de codificación depende únicamente de los patrones presentes en los datos de la base de datos original (y con ello de qué datos estén presentes en la base de datos), pues la forma en la que se obtienen los códigos busca preservar dichos patrones.

4.4.1. Generalidades

La idea detrás del CENG es conceptualmente simple: se busca encontrar el mapeo de cada instancia categórica de una base de datos a un número estandarizado entre cero y uno, de tal forma que los patrones presentes en la base de datos se preserven.

Los patrones que se busca preservar pueden ser extremadamente complejos y resulta ingenuo elegir un modelo único para capturar las relaciones que hay entre las diferentes variables presentes en cualquier base de datos. Por ello, se elige un modelo que al entrenarse con distintos datos de entrada se ajusta específicamente para ese conjunto de datos: una red neuronal.

Supóngase que se tiene una base de datos con n variables (columnas) y m observaciones (renglones). Para medir qué tan buena es una codificación dada, se intenta explicar una variable en términos de las demás de la siguiente forma:

$$x_k \sim f_k(x_1, x_2, ..., x_{k-1}, x_{k+1}, ..., x_n)$$

$$\text{con } k \in \{1, 2, ..., n\}$$

A la variable x_k se le llama **variable dependiente** o **respuesta**. Al resto de las variables $x_i, i \neq k$, se les llama **variables independientes**. La función f_k es la salida de una red neuronal que obtendrá un valor de respuesta a partir de las variables independientes. Después de la fase de entrenamiento, el modelo tiene asociado un error de predicción ε_k, que mide qué tan alejada está la respuesta que arroja el modelo contra el valor en la base de datos de la variable dependiente. Éste se calcula de la siguiente forma:

$$\varepsilon_k = \max_{i \in \{1,2,...,m\}} |f_k(x_{i,1}, x_{i,2}, ..., x_{i,k-1}, x_{i,k+1}, ..., x_{i,n}) - x_{i,k}|$$

Se entrena una red neuronal de este estilo para obtener cada $f_k, k \in \{1, 2, ..., n\}$. El resultado son n redes neuronales que modelan la predicción de cada una de las n variables en términos de las otras $n - 1$. De cada uno de los errores de predicción ε_k de las n redes, se toma el máximo para definir qué tan buena o mala es la codificación dada, de la siguiente forma:

$$\varphi = \max_{k \in \{1,2,...,n\}} \varepsilon_k \tag{4.1}$$

Con esto, para una posible asignación de códigos a las variables categóricas, φ mide qué tanto se preservan o se pueden modelar las relaciones entre las variables ahí presentes. Dada esta métrica, se busca el mapeo de instancias categóricas a valores numéricos que la hace mínima; esto es, que hace que el máximo error de predicción de las n redes neuronales sea el menor posible.

Sea $\mathbb{C}$ el conjunto de todas las posibles asignaciones de números entre 0 y 1 a instancias de las variables categóricas presentes en una base de datos dada. El problema que el CENG busca resolver se puede plantear entonces como:

$$\min_{c \in \mathbb{C}} \varphi \tag{4.2}$$

o equivalentemente:

$$\min_{c \in \mathbb{C}} \max_{k \in \{1,2,\ldots,n\}} \varepsilon_k \tag{4.3}$$

Para minimizar una métrica tan compleja, la herramienta que se eligió fue un algoritmo genético. Cada individuo representa un mapeo distinto de instancias categóricas a valores numéricos, i.e. un elemento de $\mathbb{C}$. De la forma descrita en la ecuación 4.1, se puede obtener un valor para su función de adecuación φ y se buscará el individuo que la minimice, i.e. la codificación que mejor permita expresar una variable en términos de las demás. Además, se eligió medir de esta manera el error de la red y minimizarlo pues se ha visto que esta estrategia arroja una solución sobre el frente de Pareto para problemas de optimización multiobjetivo como éste, como se expuso en la sección 3.1.4. En la expresión 4.3 se puede ver claramente la estructura mín − máx del problema que resuelve el CENG.

Así, el CENG consiste de un algoritmo genético que en cada generación evalúa la función de adecuación de p individuos. Si la base de datos de entrada tiene n variables, esto equivale a entrenar y evaluar la precisión de np redes neuronales en cada generación. Con esto, se puede ver que si se trabaja con valores elevados de n y p el tiempo de ejecución del algoritmo puede dispararse fácilmente. Por ello, se busca ajustar los parámetros del CENG de tal forma que se obtenga un balance entre tiempos de ejecución razonables y una solución suficientemente buena.

4.4.2. El problema del tiempo de ejecución

El CENG es un algoritmo computacionalmente intensivo y si se utiliza ingenuamente, se pueden obtener tiempos de ejecución de varios días aún para bases de datos de tamaños reducidos. Con el fin de sacarle el mayor provecho al algoritmo, se identificaron los factores que afectaban el tiempo de ejecución y se adoptaron estrategias para obtener tiempos de ejecución razonables.

Primeramente, se realizó una implementación paralela del algoritmo descrito en [47]. Dada la estructura del algoritmo, paralelizarlo era natural. Cada individuo del algoritmo genético podría entrenar sus redes neuronales simultáneamente de forma independiente. Java provee una forma muy sencilla de hacer esto a través de la interfaz `Callable`. Una vez que el algoritmo fue paralelizado, se pudo aprovechar al máximo la capacidad de cómputo de la arquitectura sobre la que éste se ejecuta, resultando en menores tiempos

de ejecución en arquitecturas paralelas. En la figura 4.3 se pueden ver los tiempos de ejecución para una base de datos con 3 variables numéricas, 10 categóricas con 4 instancias cada una y 500 tuplas. En la tabla 4.4 se pueden ver las características de los procesadores utilizados. Para este experimento, se fijó el número de épocas por las que se entrena la red neuronal en $1,000$ y el número de generaciones del algoritmo genético en 100. Acerca de estos parámetros se hablará en la sección 4.4.3

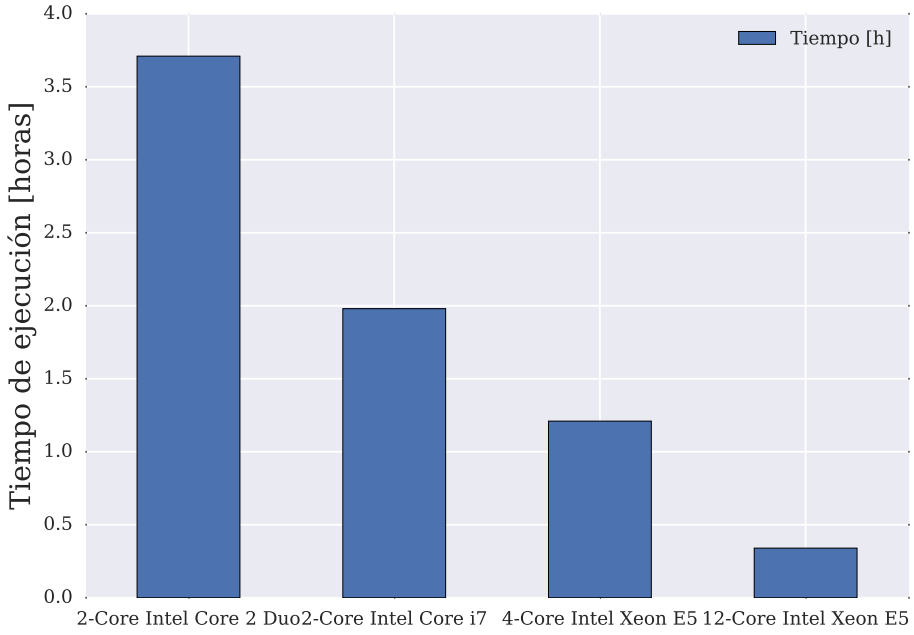


Figura 4.3. Tiempos de ejecución del CENG paralelo

En segundo lugar, el tamaño de los datos de entrada naturalmente impacta el tiempo de ejecución del CENG.

- El número de observaciones (renglones, m) en una base de datos impacta el tiempo

Procesador	**Frecuencia de reloj**	**Núcleos físicos**	**Núcleos virtuales**
Intel Core 2 Duo	2.66 GHz	2	NA
Intel Core i3	1.7 GHz	2	4
Intel Xeon E5	2.6 GHz	4	8
Intel Xeon E5	2.6 GHz	12	24

Tabla 4.4. Características de los procesadores usados

de entrenamiento de la misma, pues cada observación se utilizará para calcular el error de la red neuronal en cada pasada del algoritmo de retropropagación.

- El número de columnas (n) presentes en los datos de entrada incrementan el tiempo que tarda el CENG en encontrar una solución pues afecta la arquitectura de la red neuronal en términos del número de neuronas utilizadas en la capa de entrada. Asimismo, cada columna adicional implicará entrenar y calcular el error para una red neuronal más para cada individuo en cada generación.
- El número total de instancias categóricas (N_i) presentes en la base de datos impacta el tiempo de ejecución pues es proporcional a la longitud del genoma de los individuos y con ello tiene un efecto exponencial sobre el tamaño del espacio de búsqueda $\mathbb{C}$.

Por último, dado que el CENG consta de un algoritmo genético que entrena redes neuronales en cada iteración, se pueden ajustar varios parámetros de configuración de ambos componentes para obtener tiempos de ejecución razonables. Ejemplos de éstos son el número máximo de generaciones por las que se permite que el algoritmo genético corra y el número de épocas que se utilizan para entrenar a las redes neuronales.

El buscar un equilibrio entre una configuración que permita obtener resultados en un tiempo razonable y la precisión o calidad de éstos se convierte entonces en un problema fundamental. En la siguiente sección se explica cómo se ajustaron los parámetros del CENG para satisfacer los requisitos de tiempo de ejecución y precisión.

4.4.3. Parametrización

Existen dos conjuntos de parámetros principales que se pueden ajustar en el CENG: los correspondientes al algoritmo genético que optimizará la función objetivo y los correspondientes a las redes neuronales que se entrenan en cada generación del algoritmo genético. Ambos conjuntos de parámetros impactan tanto la precisión de la solución encontrada como el tiempo de ejecución del algoritmo. A través de los experimentos que se discuten más adelante, se determinaron los valores que debían tomar los parámetros del CENG para obtener una precisión adecuada dentro de un tiempo de ejecución razonable.

I. Número de observaciones, dimensiones e instancias categóricas en la base de datos

Como se mencionó en la sección anterior, al incrementar tanto el número de renglones como el número de columnas de la base de datos, se puede esperar un incremento lineal del tiempo de ejecución del algoritmo. El análisis anterior se corrobora en la figura 4.4, en donde se muestra el tiempo de ejecución para bases de datos con diferentes números de renglones y de columnas. Para este experimento se buscó utilizar 250 bases de datos de entre 100 y 1,000 renglones en incrementos de 100 en 100 y entre 1 y 25 columnas en incrementos de 1 en 1. El CENG se configuró para correr con 100 generaciones y 3,000 épocas para entrenar las redes neuronales. Dado que el tiempo de ejecución para las bases de datos más grandes era muy elevado y ya se podía observar una tendencia claramente lineal en los tiempos de ejecución, no se terminó de correr el experimento para los 250 ejemplos. Por ello, la segunda gráfica de la figura 4.4 presenta únicamente información para bases de datos de entre 100 y 500 tuplas.

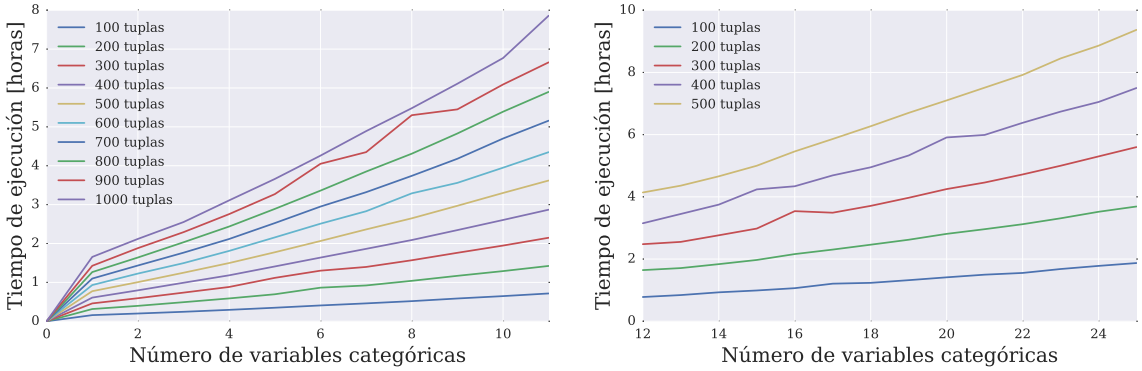


Figura 4.4. Tiempo de ejecución del EGA al variar el número de columnas y tuplas en la base de datos de entrada

Asimismo, se mencionó que incrementar el número de instancias categóricas incrementa exponencialmente el tamaño del espacio de búsqueda del genético y por ello tiene un impacto en el tiempo de ejecución. Para analizar el efecto que esto tiene sobre el tiempo de ejecución y el valor de la función de adecuación, se corrió un experimento para un conjunto de 10 bases de datos con 4 columnas categóricas y 500 tuplas, con entre 4 y 40 instancias categóricas, en incrementos de 4 en 4. Se utilizaron un máximo de 500 generaciones para el algoritmo genético y 500 épocas para entrenar la red neuronal en todos los casos. De acuerdo a los resultados que se pueden observar en la figura

4.5, se nota que el incremento en el tiempo de ejecución es prácticamente despreciable. También se puede apreciar un ligero deterioro en los valores que se obtienen de la función de adecuación. Sin embargo, es de esperarse, pues al estar fijo el número de generaciones e incrementarse el tamaño del espacio de búsqueda, el algoritmo no suele encontrar una solución tan buena como la que hubiera encontrado en un espacio de búsqueda más pequeño.

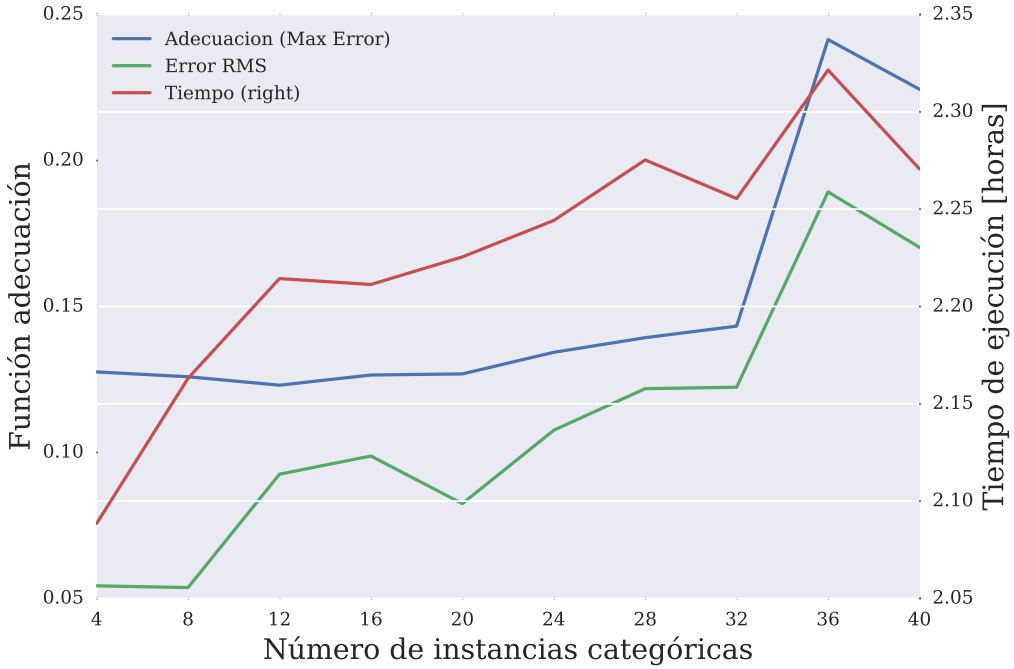


Figura 4.5. Valor de la función de adecuación y tiempo de ejecución del EGA al variar el número de instancias categóricas

II. Número de generaciones en el algoritmo genético

Se corrió el algoritmo por $1,000$ generaciones y se reportaron los valores de la función de adecuación al final de cada generación para 3 bases de datos con características diferentes. Todas ellas incluían tres variables categóricas. Sin embargo, una contenía 3 variables categóricas con 4 instancias en cada una y 700 tuplas, otra constaba de 5 variables categóricas con 4 instancias en cada una y 500 tuplas y la última tenía 9 variables categóricas con 4 instancias cada una y 300 tuplas. Se eligieron estas bases de datos particulares pues reportaron tiempos de ejecución similares al correrse utilizando

la versión secuencial del CENG.

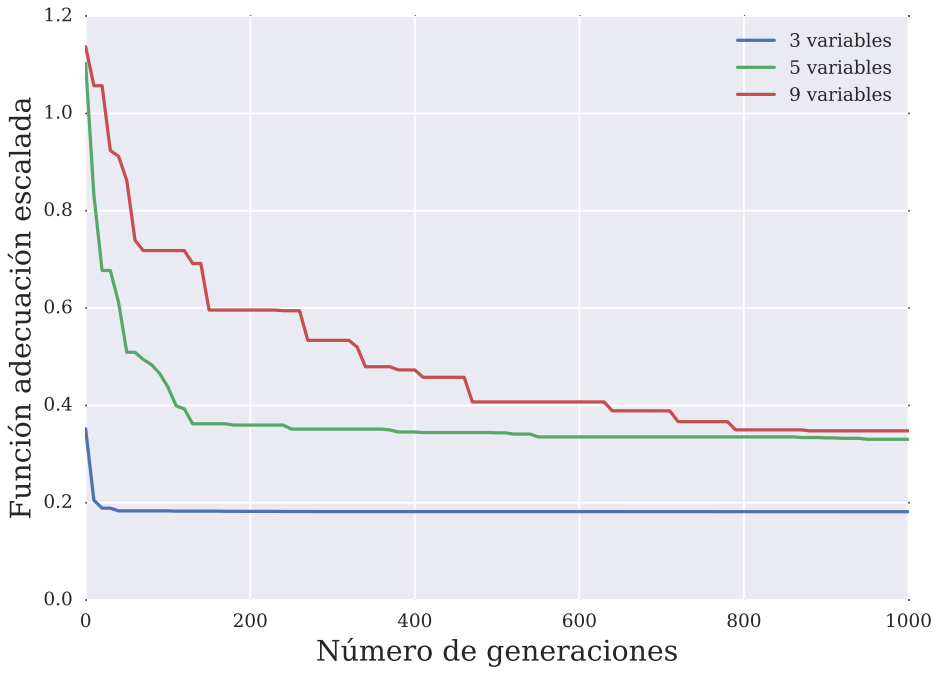


Figura 4.6. Valor de la función de adecuación del EGA al variar el número de generaciones

En la figura 4.6 se puede ver que el tamaño de la base de datos en términos de columnas y tuplas impacta directamente la velocidad de convergencia del EGA. Se hace notorio que 800 generaciones parecen ser suficientes para que el EGA converja. Sin embargo, dado que este parámetro es sensible no solo al tamaño de los datos de entrada, sino también a la complejidad de los patrones presentes en ellos, debe determinarse de forma estadística dependiendo del tipo de datos sobre los que se correrá el algoritmo.

III. Otros parámetros para el algoritmo genético

Grefenstette realizó un estudio [48] en el que busca la mejor combinación de 6 parámetros de un algoritmo genético. De éstos, 3 son de interés para nosotros. Para el algoritmo genético "estándar" Grefenstette los define de la siguiente forma:

- Número de individuos en la población $p = 50$
- Probabilidad de cruza $P_c = 0.6$: la probabilidad de que dos individuos se crucen.

- Probabilidad de mutación $P_m = 0.001$: la probabilidad con la que un bit del genoma se altera.

En su estudio utiliza un "meta-algoritmo genético" que modifica los parámetros del algoritmo genético que se está usando para resolver el problema de interés. Para uno de sus experimentos concluye que el algoritmo genético que mejor desempeño tiene, se configura con los parámetros:

- Número de individuos en la población $p = 30$
- Probabilidad de cruza $P_c = 0.95$
- Probabilidad de mutación $P_m = 0.01$

En particular, en el CENG, se configuraron estos parámetros de la siguiente manera:

- Número de individuos en la población $p = 40$
- Probabilidad de cruza $P_c = 1$
- Probabilidad de mutación $P_m = 0.05$

Una probabilidad de cruza tan elevada introducirá más variedad en la población, lo cual resulta importante pues se trabaja con un número reducido de individuos. Por la misma razón, la probabilidad de mutación tan elevada en comparación con el algoritmo genético estándar que expone Grefenstette ayuda a prevenir convergencia prematura a óptimos locales.

Además de estos parámetros se podría tomar en cuenta otros que modifican las características estructurales del genético, como la estrategia de selección o de cruza. En la sección 3.1.2 se menciona el estudio de Kuri [21, 22] en donde se compara el desempeño de algoritmos genéticos estructuralmente distintos. En este trabajo se utiliza el algoritmo genético ecléctico (EGA) que resultó ser el mejor en dicho estudio.

IV. Número de épocas en las redes neuronales

Para determinar el número de épocas necesarias para entrenar las redes presentes en CENG, se corrió el algoritmo con una configuración fija y variando únicamente el número de épocas que se utilizaban para entrenar las redes neuronales. Se configuró el CENG para correr por un máximo de 100 generaciones y se le alimentaron bases de datos que consistían de 3 variables numéricas y 10 variables categóricas con 4 instancias cada

una. Se ejecutó entonces el CENG variando el número de épocas desde 20 hasta 500 en incrementos de 20 épocas. Cada una de estas configuraciones se ejecutó sobre 10 bases de datos con las características antes mencionadas pero con un número de tuplas que variaba de 100 a 1000 en incrementos de 100. Los resultados de este experimento se pueden ver en la figura 4.7.

Figura 4.7. Valor de la función de adecuación del EGA al variar el número de épocas usadas para entrenar las redes neuronales

Después de las 200 épocas, parece no haber una mejora significativa en el valor de la función de adecuación final reportada por el EGA. Se exploró también el efecto que tenía correr el algoritmo utilizando más de 500 épocas de entrenamiento y se llegó a la conclusión de que no se obtenía ninguna mejora adicional. Por ende, se concluyó que no se requieren más de 200 épocas para entrenar las redes neuronales del CENG para que modelen los patrones presentes en la base de datos de entrada. Sin embargo, como en el caso del número de generaciones, se cree que esta configuración depende de los datos de entrada y la complejidad de los patrones presentes en ellos. Por esta razón, se recomienda determinar este parámetro de forma estadística utilizando datos similares a los que se emplearán en cualquier otra aplicación del CENG.

V. Otros parámetros para la red neuronal

Vale la pena describir también algunos otros parámetros que no se tomaron en cuenta para el análisis anterior como son la arquitectura tiene la red neuronal que se está utilizando y la función de activación que utilizan las neuronas. También se mencionará el algoritmo que se utiliza para entrenar la red neuronal, aunque no sea propiamente un parámetro pues no depende de los datos.

Como se mencionó en la sección 3.3.2, Cybenko [30] mostró que sólo se requiere una capa de neuronas ocultas en una red neuronal para aproximar cualquier función continua en el hipercubo unitario. Como se puede ver en la figura 3.4, para definir la arquitectura de una red de perceptrones con una sola capa oculta, basta con definir el número de neuronas de la capa oculta. La capa de entrada tendrá tantas neuronas como entradas tenga la función a aproximar. La capa de salida constará siempre de una neurona para obtener el valor de la imagen de la función aproximada. En el CENG el número de neuronas en la capa oculta está determinada por la cota de la que se habló al final de la sección 3.3.2.

El teorema 3.5 de Cybenko es válido para la familia de funciones de activación continuas, monótonas crecientes y acotadas. En particular, en el CENG las neuronas utilizan $f(x) = \tanh(x)$ como función de activación, pues satisface las condiciones del teorema y es ampliamente utilizada.

Por último, la red neuronal está entrenada utilizando el algoritmo de retropropagación [29] que itera sobre la red ajustando los pesos de las conexiones entre neuronas de acuerdo a los errores de predicción medidos sobre un conjunto de datos de entrenamiento.

Nótese que el anásis descrito en las secciones 4.4.2 y 4.4.3 permite justificar la elección de los parámetros con los que se ejecutó el CENG. Los resultados de este análisis también se comentan en [49]. Se pretendió limitar su tiempo de ejecución en bases de datos de tamaño similar a las que se utilizaron en los experimentos principales sin sacrificar calidad en la codificación resultante.

4.5. Estructura de la base de datos

Con base en las frecuencias reportadas en las tablas 4.2 y 4.3 y a los resultados obtenidos en la sección 4.4.2 se optó por combinar algunas de las categorías gramaticales en una sola variable categórica que tendría como instancias las etiquetas posibles para todas las categorías gramaticales que incluye. También, para categorías gramaticales que resultan ser muy comunes en la muestra de textos analizados, se considera que podrían aparecer dos veces en el modelo de oración que se propondrá.

En la tabla 4.5 se pueden ver las categorías gramaticales que se tomarán en cuenta para el análisis del texto en cada variable categórica de la base de datos, cuántas variables de ese tipo habrá y cuántas instancias existirán en cada una de ellas.

Tomando en cuenta los datos de las tablas anteriores y algunos otros marcadores estilísticos mencionados en la sección 2.3, se definió la estructura ilustrada en la tabla 4.6 para modelar el texto. La idea básica detrás del modelo es que una oración debe contener al menos uno de cada uno de los elementos gramaticales mostrados en la tabla. Si bien esto no es cierto desde un punto de vista lingüístico pues el lenguaje es demasiado complejo como para encajonarse en una estructura rígida, sirve como una aproximación que se espera preserve la suficiente información para identificar el estilo con el que se escribió.

Dado que garantizar que cada oración analizada contiene todos los elementos de la tabla 4.6 es infactible, se agrega una instancia adicional a cada categoría: un valor nulo. La presencia de esa instancia en una oración implica que para dicha oración, no se encontró una palabra que corresponda a esa categoría. Así, el número de instancias totales se incrementa en 10 (el número de variables categóricas presentes en la base de datos) al tomar en cuenta los valores nulos.

Ahora bien, como es ingenuo suponer que cada oración gramatical podrá encajonarse en esta estructura rígida que pretende modelar una oración, se define el concepto de ***fraze***. Una *fraze* es una secuencia de palabras modelada por la estructura definida en la tabla 4.6. Ésta puede corresponder a una oración gramatical, a una parte de una oración gramatical, o a varias oraciones gramaticales.

Nótese que, además de las variables categóricas descritas anteriormente, se incluyen en la estructura tres variables numéricas que modelan marcadores estilísticos léxicos

Primera letra del tag	Número de categorías	Número de instancias
A	1	2
C	1	2
D	1	7
F	1	15
N	2	5
P	1	9
R,S	1	3
V	1	33
I,W,Z	1	4
3	0	0
Total	10	80

Tabla 4.5. Elección de categorías y número de instancias para el modelo de oración

Variable	Tipo	Instancias	Descripción
`A`	categórica	3	Adjetivos
`C`	categórica	3	Conjunciones
`D`	categórica	8	Determinantes
`F`	categórica	16	Signos de puntuación
`N1`	categórica	6	Sustantivos
`N2`	categórica	6	Sustantivos
`P`	categórica	10	Pronombres
`RS`	categórica	4	Adverbios y preposiciones
`V`	categórica	34	Verbos
`IWZ`	categórica	5	Interjecciones, fechas y números
`_tokens`	numérica	-	Número de tokens analizados
`_words`	numérica	-	Número de palabras analizadas
`_punct`	numérica	-	Número de signos de puntuación analizados

Tabla 4.6. Modelo de oración: *fraze*

utilizados en los estudios mencionados en la sección 2.3.

La base de datos con esta estructura se irá llenando analizando palabra por palabra del texto a modelar, y registrando en una *fraze* la instancia correspondiente para la categoría de la palabra en cuestión. Si para una categoría de la *fraze* ya se había analizado anteriormente una palabra de la misma categoría, la segunda aparición es ignorada. Así, palabra a palabra se va "llenando" cada *fraze*. El criterio de corte entre *fraze* y *fraze* se definió buscando garantizar alguna de las siguientes dos condiciones:

- Que cada *fraze* no contenga un número muy elevado de valores nulos.
- Que no se analice un número muy elevado de tokens para cada *fraze*.

Para definir valores para estos criterios, se hizo un análisis estadístico sobre el corpus descrito en la sección 4.2. Primero que nada, se llenó cada *fraze* con una oración gramatical. Para determinar cuándo termina una oración gramatical, se buscó un punto o un signo de exclamación o interrogación de cierre. Para cada *fraze* llenada de esta manera, se midió el número de tokens analizado y el número de nulos presentes en la base de datos y se obtuvieron los resultados mostrados en la tabla 4.7.

A partir de esto, se decidió fijar primero el número de nulos presentes en cada *fraze*. Suponer normalidad en la distribución de los datos no era factible. Utilizar la cota de Chebyshev para garantizar que se cubría cierto porcentaje de los casos resultaba en una cota demasiado laxa que hubiera implicado perder mucha de la información presente en el texto (tomar 5.8 valores nulos como criterio de corte para cubrir el 75 % de los casos [$k = 2$ en la desigualdad de Chebyshev]). Por ello, se optó por simplemente utilizar el tercer cuartil muestral como criterio de corte, i.e. el 75 % de las observaciones de la

Medida	**Número de nulos**	**Número de tokens**
Min	0.00	1.00
Q1	1.00	12.00
Q2	2.00	22.00
Media	2.31	28.41
Q3	3.00	35.00
Max	9.00	496.00
σ	1.75	26.97

Tabla 4.7. Análisis por oraciones gramaticales

muestra tenían un número de nulos menor o igual a 3.

Al cambiar el criterio de corte de signos de puntuación a 3 valores nulos por *fraze*, se obtuvieron los resultados de la tabla 4.8. Para acotar en rango y la variabilidad del número de tokens analizados, se optó nuevamente por elegir el tercer cuartil como criterio de corte. Así, el límite de una *fraze* está definido por lo que ocurra primero:

- Que el número de nulos en la *fraze* sea 3 o menor.
- Que el número de tokens analizados para llenar una *fraze* sea mayor a 12.

Medida	**Número de nulos**	**Número de tokens**
Min	3	7.00
Q1	3	9.00
Q2	3	10.00
Media	3	11.07
Q3	3	12.00
Max	3	66.00
σ	0	3.68

Tabla 4.8. Análisis usando 3 o menos nulos por *fraze* como criterio de corte

En la tabla 4.9 se pueden ver los estadísticos resultantes de llenar una base de datos utilizando estos dos criterios de corte sobre todo el corpus de prueba.

Así, se puede ver claramente como una *fraze* no necesariamente modela una oración, sino una secuencia de palabras en el texto que satisface los criterios definidos anteriormente. En esta sección se presentó sólo una forma de definir la estructura de la base de datos, pero está claro que esto se podría hacer de diversas maneras. En trabajo futuro se podría experimentar con otras formas de procesar el texto original para llevarlo a un formato que sea aceptado por el CENG.

4.6. Procesamiento numérico

Existen diversos métodos de aprendizaje no supervisado que se podrían utilizar para detectar los patrones numéricos presentes en la base de datos numérica resultante del

Medida	Número de nulos	Número de tokens
Min	3.00	7.00
Q1	3.00	9.00
Q2	3.00	10.00
Media	3.22	10.32
Q3	3.00	13.00
Max	7.00	13.00
σ	0.50	2.04

Tabla 4.9. Análisis usando 3 o menos nulos por *fraze* o más de 12 tokens analizados como criterio de corte

CENG. Para este trabajo se eligieron dos algoritmos de agrupamiento muy distintos, para poder comparar su desempeño. En primera instancia, se eligió la herramienta de agrupamiento entrópico [39] descrita en la sección 3.4.2. Dicha herramienta es extremadamente robusta y por la peculiaridad de la métrica que utiliza para generar los clusters podría ser capaz de capturar algunos patrones que pasan desapercibidos otras herramientas. Sin embargo, su ejecución es tardada, pues se basa en un algoritmo genético y no resulta práctica para todo el proceso, como se verá en la sección 5.1. Aún así, sí se reportan resultados para la similitud entre textos utilizando este algoritmo.

Adicionalmente, se optó por agrupar los datos utilizando mapas auto-organizados pues son una herramienta robusta con un menor costo computacional que el agrupamiento entrópico. Como se mencionó en la sección 3.4.3, el modelo de aprendizaje por refuerzo y competencia de los mapas de Kohonen permite detectar cúmulos con formas más complejas a las que tendrían los cúmlos encontrados por un algoritmo que trabajara únicamente calculando promedios (como k-medias). Si se utilizan tantos nodos en la retícula de un mapa auto-organizado como cúmulos se desea encontrar, el resultado de los pesos para cada nodo funciona a manera de centroide de un cúmulo y sirve para etiquetar los datos originales. Así, el BMU sobre la retícula resultante para cada vector de entrada, determina la etiqueta de la clase a la que pertenece ese vector.

Aunque la forma en la que funcionan los dos algoritmos es distinta, ambos cumplen el propósito de un algoritmo de agrupamiento: generar un *vector de asignación* para la base de datos sobre la que se entrenan.

Definición 4.1. Sea n el número de observaciones en un conjunto de datos numéricos. El **vector de asignación** para el conjunto de datos dado un algoritmo de agrupamiento es un vector

$$a = [a(1), a(2), ...a(n)]$$

donde cada entrada $a(i) = j$ si la i-ésima observación fue asignada al cúmulo j por el algoritmo de agrupamiento con $j \in \{1, 2, ..., k\}$ y $k =$ el número de cúmulos para el que se corrió el algoritmo de agrupamiento.

El concepto de vector de asignacion se utilizará en la siguiente sección para definir algunas métricas de similitud entre distintos agrupamientos.

4.7. Mediciones de similitud

Como se vio en la sección anterior, se podrían elegir diversas herramientas de procesamiento numérico para intentar extraer los patrones presentes en la base de datos. En el caso de este trabajo, una vez que se genera un agrupamiento para la base de datos que representa el texto de un autor, se debe definir una métrica de similitud. De forma intuitiva, dicha métrica permitirá identificar que tan parecidos son dos textos, para finalmente poder decir si pertenecen al mismo autor.

Dada la elección de algoritmos de procesamiento numérico explicada en la sección anterior, la métrica debe cuantificar la similitud entre los agrupamientos generados por el mismo algoritmo para dos textos distintos. Nótese que siempre que se mide la similitud entre textos, éstos deben haber sido procesados con **el mismo** algoritmo de agrupamiento. Podría haber muy distintos enfoques para definir esta métrica. En este trabajo se definirán tres de ellas, ilustrando la intuición detrás de su construcción.

I. Probabilidad de coincidencia

Primeramente, se optó por analizar el número de veces en los que para dos bases de datos coincide la asignación de los puntos a los cúmulos. Partiendo del supuesto que existen el mismo número de cúmulos k en ambas bases de datos, aparece un problema combinatorio inicial. Dados dos grupos de k cúmulos resultantes, nada garantiza que las etiquetas de ambos grupos coincidan. Esto es, el cúmulo 1 del grupo 1 no tiene por

qué ser el cúmulo etiquetado como 1 en el segundo grupo. Un ejemplo claro de esto se puede ver en la figura 4.8, en dónde se muestran dos posibles formas de etiquetar los cúmulos del famoso conjunto de datos de Iris de Fisher [50].

Nótese que para k cúmulos, existen $k!$ formas de etiquetarlos. Entonces el primer problema será encontrar la manera en la que las etiquetas del primer grupo de cúmulos se podrían mapear al segundo. Una forma ingenua de hacer esto es fijar las etiquetas de uno de los grupos y explorar todas las posibles asignaciones para el otro. Esto resulta en explorar $k!$ posibilidades, lo cual escala muy mal con el valor de k.

Suponiendo inclusive que se exploraran todas las posibilidades, ¿cómo se puede identificar que se ha encontrado la forma de etiquetar al segundo grupo que coincide con la manera en la que está etiquetado el primero? Se optó entonces por definir que el mapeo de etiquetas correcto sería aquél que maximizara el número de coincidencias en los cúmulos. Esto podría no ser cierto; sin embargo, permite medir el mejor de los casos, i.e. la posibilidad en la que podrían parecerse más dos textos.

Con tal de no explorar las $k!$ posibilidades, se decidió explorar una muestra aleatoria de tamaño mín$\{k!, 1000\}$ del conjunto de todas las posibles permutaciones. Para cada una de esas asignaciones, se contaron el número de puntos que fueron agrupados en el mismo cúmulo en ambas bases de datos. Se toma de éstas, la asignación que maximiza el número de coincidencias.

Será entonces el número de coincidencias el que permite definir qué tan similares son dos textos. De forma intuitiva, cuantos más puntos se hayan agrupado bajo el mismo cúmulo en ambas bases de datos, más similares serán los dos textos. Para cuantificar de

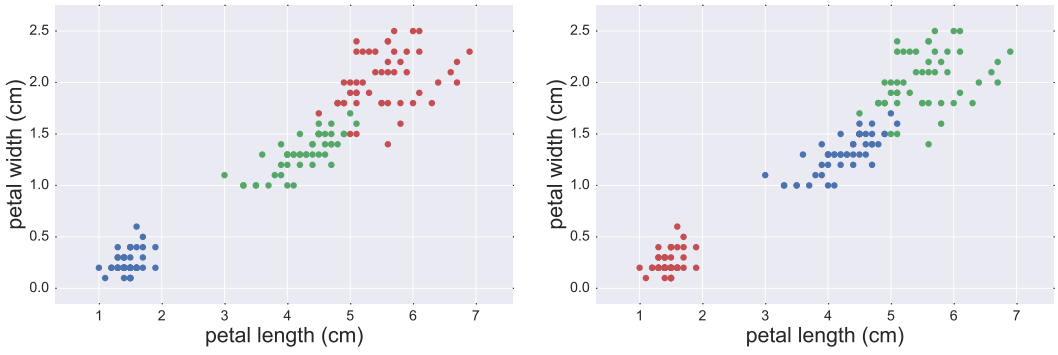


Figura 4.8. Dos ejecuciones de un algoritmo de agrupamiento pueden generar los mismos cúmulos etiquetados de diferente manera

forma más evidente cuántas coincidencias son suficientes para decir que los dos textos son similares, se hizo un análisis para identificar la probabilidad de que dos distintas formas de etiquetar una base de datos generen al menos m coincidencias.

Dados T elementos que se van a agrupar en k cúmulos, se tienen k^T posibles formas de agruparlos. La pregunta a resolver es: ¿cuántas de estas formas coinciden en exactamente m posiciones? El problema equivale a asignar en T posiciones, una de k etiquetas Si se fijan m posiciones que son las que coinciden con el resto:

$$\binom{T}{m}$$

y se obliga a que las restantes $T - m$ posiciones **no** coincidan con la otra forma de etiquetar los datos

$$(k-1)^{T-m}$$

Entonces el número de asignaciones que coinciden en exactamente m posiciones es:

$$\binom{T}{m}(k-1)^{T-m}$$

Si se desea obtener el número de asignaciones que coinciden en **al menos** m posiciones, basta con sumar

$$\sum_{i=m}^{T}\binom{T}{i}(k-1)^{T-i}$$

Y para calcular la probabilidad de que haya al menos m coincidencias:

$$\frac{\sum_{i=m}^{T}\binom{T}{i}(k-1)^{T-i}}{k^T} \tag{4.4}$$

Nótese que si se suma sobre todos los valores de i, se obtiene

$$\frac{\sum_{i=0}^{T}\binom{T}{i}(k-1)^{T-i}}{k^T} = 1$$

mostrando que en efecto se están considerando todas las posibilidades.

En el contexto de bases de datos con *frazes* para cada texto, se tienen las siguientes definiciones:

Sea T el número de *frazes* presentes en un texto. Se puede reescribir la definición 4.1 como sigue para adaptarse a la interpretación de que la base de datos usada modela un texto.

Definición 4.2. El **vector de asignación** para un texto dado un algoritmo de agrupamiento es un vector

$$a = [a(1), a(2), ...a(T)]$$

donde cada entrada $a(i) = j$ si la i-ésima *fraze* fue asignada al cúmulo j por el algoritmo de agrupamiento con $j \in \{1, 2, ..., k\}$ y $k =$ el número de cúmulos para el que se corrió el algoritmo de agrupamiento.

Definición 4.3. El **número de coincidencias** entre el texto i y el texto j es

$$m_{ij} = \text{num\_zero}(a_i - a_j)$$

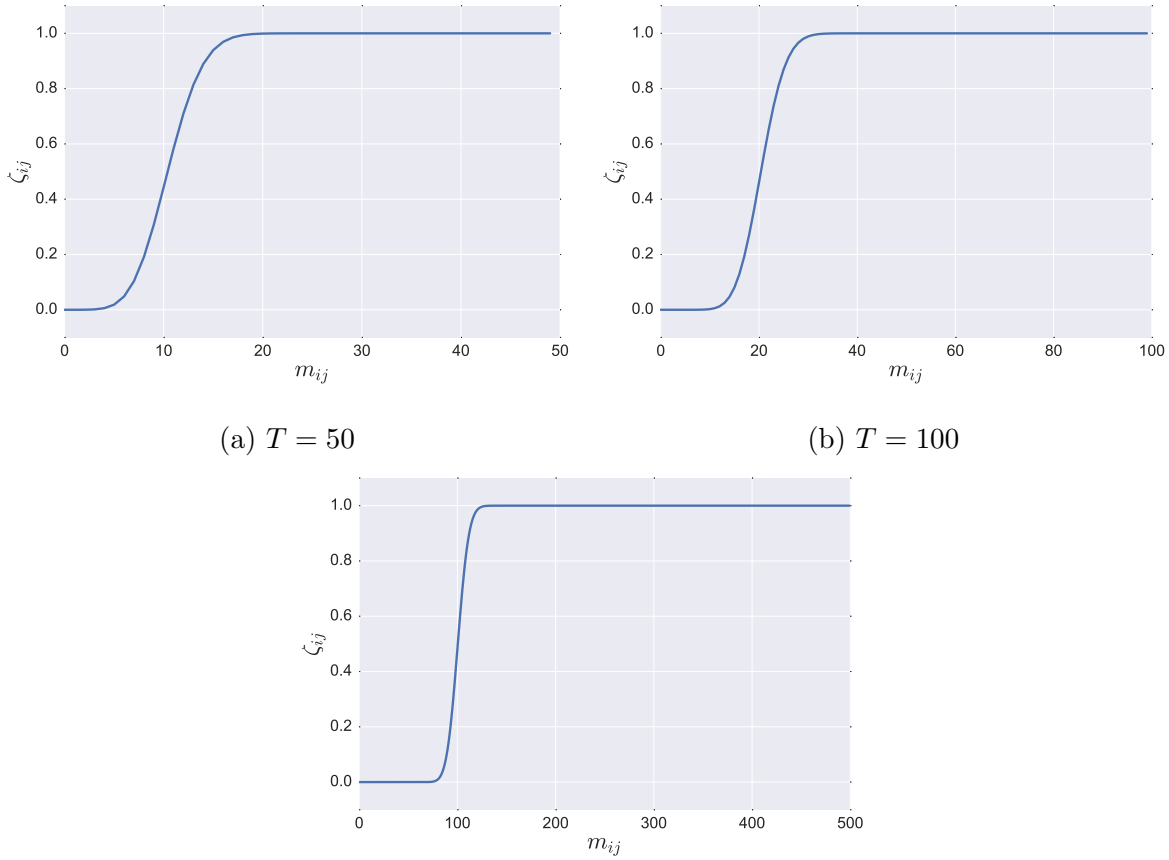
dónde la función num_zero(x) cuenta el número de entradas que son cero en el vector x y a_i, a_j son los vectores de asignación (definición 4.2) para las bases de datos que modelan al texto i y al texto j dado algún algoritmo de agrupamiento.

Definición 4.4. Se define la **similitud por probabilidad de número de coincidencias** entre el texto i y el texto j como

$$\zeta_{ij} = 1 - \frac{\sum_{l=m_{ij}}^{T} \binom{T}{i} (k-1)^{T-l}}{k^T}$$

con m_{ij} el número de coincidencias entre el texto i y el texto j.

Así, se tiene una medida que cuantifica qué tan probable es que el número de coincidencias entre dos bases de datos para cada uno de los cúmulos sea al menos m_{ij}. Cuanto menos probable sea que dos formas de etiquetar los cúmulos cualquiera coincidan en ese número de valores, se puede decir que las bases de datos son más similares. Esto generará valores de ζ_{ij} más cercanos a uno. Cuanto más probable sea que una forma de etiquetar los cúmulos tenga al menos ese número de coincidencias, se puede decir que las bases de datos son más distintas. Esto generará valores de ζ_{ij} más cercanos a cero.



(a) $T = 50$

(b) $T = 100$

(c) $T = 500$

Figura 4.9. Similitud por probabilidad de número de coincidencias ζ_{ij} con $k = 5$

II. Número de coincidencias

Como se puede ver en la figura 4.9, para valores muy bajos o muy altos de m_{ij}, ζ_{ij} pierde la capacidad de distinguir diferencias entre textos. Por ello, se decidió analizar también simplemente el valor de m_{ij}. Dado que ζ_{ij} es una función monótona creciente de m_{ij}, es posible que simplemente analizar m_{ij} nos permita distinguir de forma clara la similitud entre textos.

Definición 4.5. Se define la **similitud por número de coincidencias** entre el texto i y el texto j como m_{ij}

III. Distancia entre vectores de asignación

Otra posible forma de medir la similitud entre agrupaciones sería utilizando las proporciones en las que fueron asignadas las *frazes* a cada cúmulo.

Definición 4.6. El **vector de proporciones de asignación** para un texto dado un algoritmo de agrupamiento es un vector

$$p = [p(1), p(2), ..., p(k)]$$

donde cada entrada $p(j)$ contiene el número de *frazes* que fueron asignadas al cúmulo j.

Definición 4.7. Se define la **similitud por proporciones** entre el texto i y el texto j como

$$\eta_{ij} = ||p_i - p_j||_2$$

Nótese que esta forma de medir similitud, a diferencia de las dos métricas anteriores resulta inversamente proporcional a la similitud entre los textos. Esto es, dos textos idénticos tendrán $\eta_{ij} = 0$, mientras que dos textos muy distintos tendrán valores elevados de η_{ij}.

Claramente, para esta métrica existe el mismo problema combinatorio del etiquetado de cúmulos discutido anteriormente. Por ello, se analizarán un máximo de mín$\{k!, 1000\}$ posibles permutaciones de etiquetas y se tomará el mínimo valor de η_{ij} encontrado.

4.8. Comentarios sobre el procesamiento numérico

Es importante resaltar que las últimas dos secciones presentan sólo una forma de procesar los datos numéricos resultantes de CENG. La ventaja principal de CENG es que la base de datos numérica resultante se podría procesar de la manera que se desee. Nótese que la definición de la métrica de similitud expuesta en esta sección depende del algoritmo numérico elegido para identificar los patrones presentes en el conjunto de números. Poder variar dicho algoritmo numérico y con él, la métrica de similitud, genera un sinfín de posibilidades en las que se podría llevar a cabo el análisis de los resultados del CENG. Analizar más posibilidades está fuera de los límites de este trabajo y presenta una línea sobre la que se podría investigar en el futuro.

5. Experimentos y Resultados

> *Remember that all models are wrong; the practical question is how wrong do they have to be to not be useful.*
>
> – George Box

Para poner a prueba el funcionamiento del algoritmo propuesto, se eligieron tres conjuntos de datos con los cuáles experimentar. La diferencia principal entre los conjuntos de datos era la longitud de los textos analizados, como se verá más adelante. Posteriormente, para tener un poco más de claridad sobre los resultados, se decidió correr experimentos sobre un conjunto de datos adicional, usado anteriormente por Ángel Kuri en el trabajo que motivó esta tesis [1].

5.1. Experimentos

I. Determinación del número de cúmulos

Determinar el número de cúmulos presentes en un conjunto de datos, es un problema abierto de aprendizaje no-supervisado. No existe una forma exacta de calcularlo y muchos algoritmos de agrupamiento lo requieren como un parámetro de entrada, aunque existen algoritmos que no lo requieren como DBSCAN, OPTICS y los algoritmos jerárquicos de agrupamiento. En la práctica, existen una variedad de soluciones que funcionan, sin embargo, entrar en el detalle de todas ellas se escapa de los límites de este trabajo. Una buena aproximación se puede obtener usando el método del codo, propuesto originalmente por Thorndike [51]. Dicho método consiste en visualizar algún índice de calidad del agrupamiento generado para k cúmulos para diferentes valores consecutivos de k. Las métricas de calidad suelen ser inversamente proporcionales al número de cúmulos. El método dice que si se observa la tendencia de los datos, para

cierto valor de k se puede distinguir un quiebre (o codo) en la gráfica. Dicho valor de k es el número de cúmulos con el que se debe correr el algoritmo de agrupamiento.

Para cada uno de los conjuntos de datos se presentan algunas de las gráficas que se utilizaron para determinar el número de cúmulos presentes en los datos. El índice de calidad elegido para cada agrupamiento generado fue la suma de las distancias al cuadrado de cada punto al centro del cúmulo al que pertenece. Dicho índice de calidad se conoce en inglés como "within-cluster sum of squares" (WCSS) y se define como

$$WCSS = \sum_{j=1}^{k} \sum_{x \in C_j} ||x - c_j||^2 \tag{5.1}$$

en donde k es el número de cúmulos, c_j representa el centroide del cúmulo j y la pertenencia de un punto x al cúmulo C_j está determinada por el algoritmo de agrupamiento usado.

En ocasiones el codo en la gráfica puede ser muy sutil. Por ello, para detectarlo con mayor facilidad se suele graficar no sólo el índice de calidad para un agrupamiento con k cúmulos, si no el cambio en el índice de calidad entre un agrupamiento para k cúmulos y uno para $k-1$ cúmulos. En este caso, se presentan dos gráficas, una que muestra el $WCSS$ y otra que muestra

$$WCSS_{k-1} - WCSS_k$$

para distintos valores de k.

II. Algoritmos de agrupamiento

Se probaron los dos algoritmos de agrupamiento distintos descritos en la sección 4.6. En las siguientes tablas, SOMS hace referencia al agrupamiento utilizando mapas autoorganizados y CBE al agrupamiento basado en entropía. Para determinar el número de cúmulos presentes en los datos se utilizó únicamente el algoritmo de mapas autoorganizados, pues el agrupamiento entrópico es computacionalmente intensivo y hubiera sido necesario correrlo cerca de 40 veces para generar cada una de las gráficas para detectar el número de cúmulos. Sin embargo, se compararon los resultados de similitud generados al utilizar ambos algoritmos de agrupamiento.

III. Resultados de similitud

Adicionalmente, una vez que se determinó el número de cúmulos presentes en los datos del conjunto de pruebas y se ejecutó el algoritmo de agrupamiento, se evaluó la similitud entre textos usando las tres métricas explicadas en la sección 4.7.

Dado que se cuenta con un número elevado de textos en la mayoría de los experimentos y la similitud está definida para comparar sólo dos textos a la vez, se organizan los resultados de similitud en matrices de la forma

$$Z_{T\times T} = (\zeta_{ij}),\ M_{T\times T} = (m_{ij}) \text{ y } H_{T\times T} = (\eta_{ij}) \tag{5.2}$$

donde la entrada ij de cada matriz corresponde a la medición de similitud entre el texto i y el texto j y T es el número de textos analizados en esa muestra (ver definiciones 4.4, 4.5 y 4.7).

Aunque las tres diferentes mediciones de similitud se encuentran en diferentes rangos, se presenta un mapa de calor para cada una de las matrices, pues permite visualizar con facilidad en qué rango se encuentran los valores. El código de color para los mapas de calor ha sido elegido de tal forma que siempre se visualice en azul un valor que representa alta similitud entre textos y en rojo un valor que representa baja similitud. Además, se presenta para cada experimento también el mapa de calor de la matriz indicadora

$$A_{T\times T} = (a_{ij}) \text{ en donde } \begin{cases} a_{ij} = 1 & \text{si el texto } i \text{ y el texto } j \text{ son del mismo autor} \\ a_{ij} = 0 & \text{si el texto } i \text{ y el texto } j \text{ son de autores distintos} \end{cases} \tag{5.3}$$

Así, se pueden comparar los mapas de calor con el patrón que deberían presentar si la métrica de similitud permitiera detectar la diferencia entre autores.

5.1.1. Textos con 500 frazes

A partir del corpus de textos descrito en la sección 4.2, se generaron textos con *frazes* como las definidas en la sección 4.5. De la muestra de textos, sólo se incluyeron en este experimento los textos que generaban bases de datos con al menos 500 *frazes*.

Posteriormente se truncaron estas bases de datos a exactamente 500 *frazes* cada una, para hacer posible el análisis de similitud, dado que se basa en medir el número de coincidencias en la asignación de cúmulos.

Así, para este experimento se analizaron 17 textos de 7 autores distintos.

I. Determinación del número de cúmulos

En las gráficas de la figura 5.1 se pueden ver los $WCSS$ para algunos de los textos de la muestra:

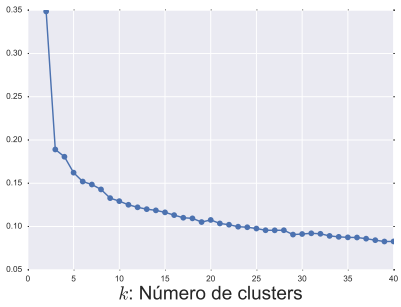
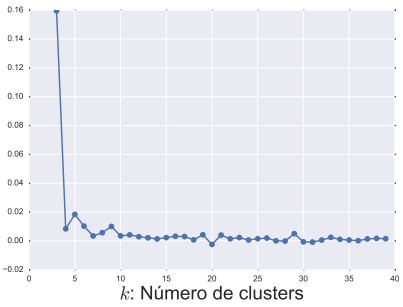
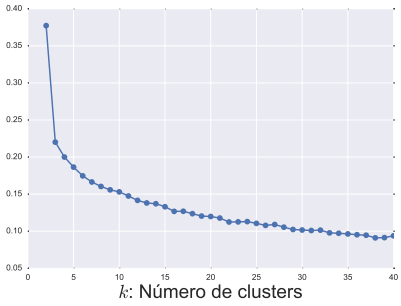
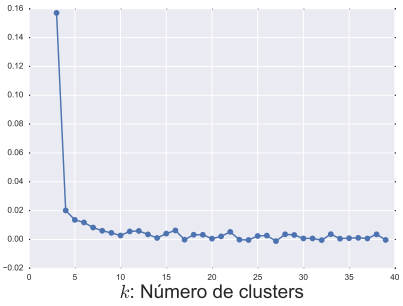
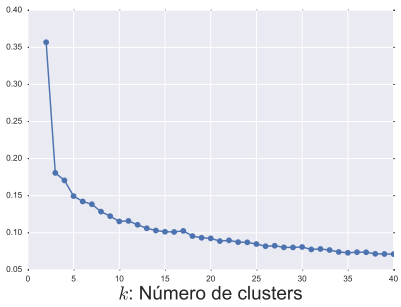
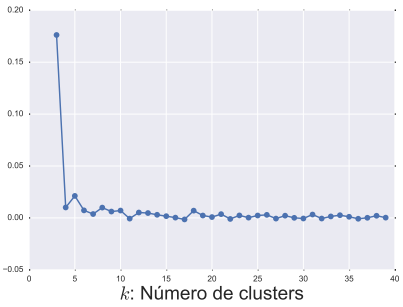
Índice de calidad / Texto	$WCSS_k$	$WCSS_{k-1} - WCSS_k$
Texto 1		
Texto 2		
Texto 3		

Figura 5.1. Gráficas del índice de calidad para textos con 500 *frazes* codificados con CENG

Esto permitió concluir que se utilizaría una $k = 5$ para un número de cúmulos.

II. Resultados de similitud

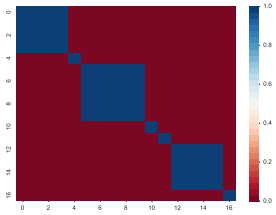


Figura 5.2. Mapa de calor de la matriz A para textos con 500 *frazes*

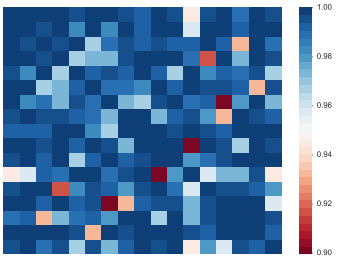
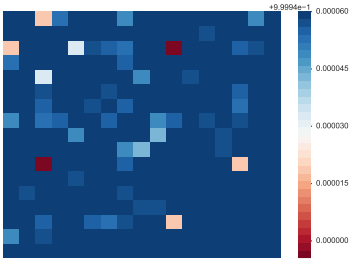
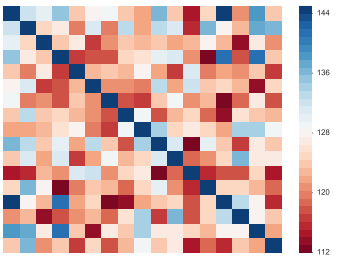
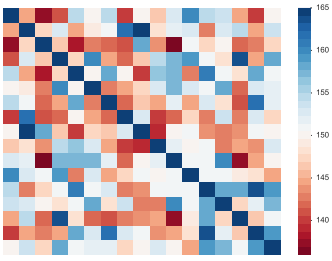
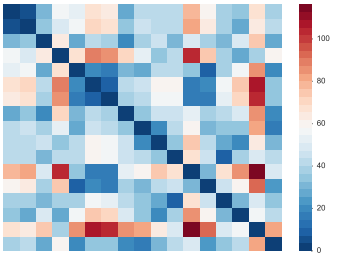
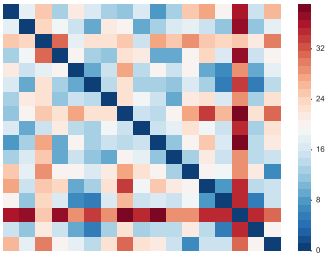
Métrica \ Clustering	SOMS	CBE
Similitud por probabilidad de número de coincidencias: $Z = (\zeta_{ij})$		
Similitud por número de coincidencias: $M = (m_{ij})$		
Similitud por proporciones: $H = (\eta_{ij})$		

Figura 5.3. Mapas de calor de las matrices de similitud para textos con 500 *frazes* codificados con CENG

5.1.2. Textos con 100 frazes

Se utilizó el mismo enfoque que en el experimento anterior para generar las bases de datos. Para este experimento se tomaron en cuenta textos que generaban bases de datos de al menos 100 *frazes*. Así, se analizaron 34 textos de 10 autores distintos.

I. Determinación del número de cúmulos

En las gráficas de la figura 5.4 se pueden ver los $WCSS$ para dos textos de la muestra:

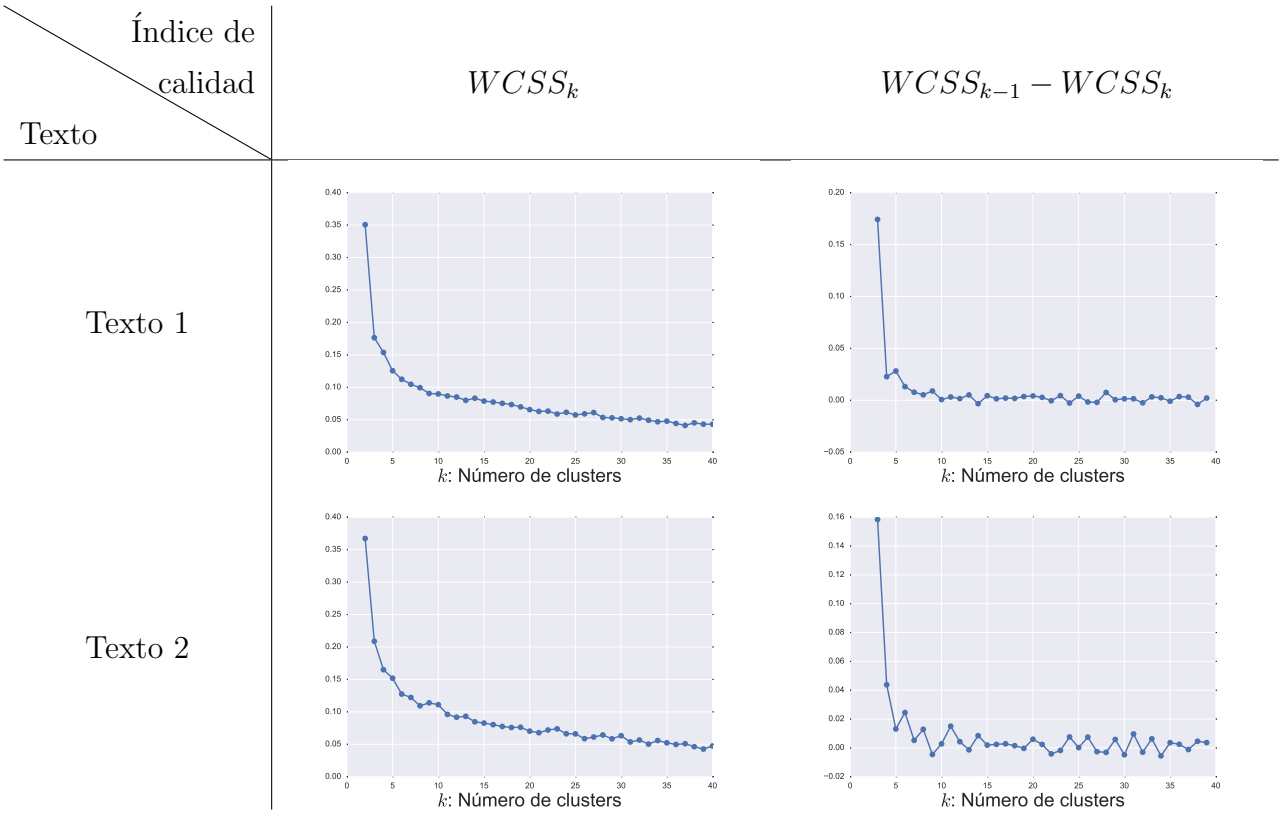


Figura 5.4. Gráficas del índice de calidad para textos con 100 *frazes* codificados con CENG

Esto permitió concluir que se utilizaría una $k = 5$ para un número de cúmulos.

II. Resultados de similitud

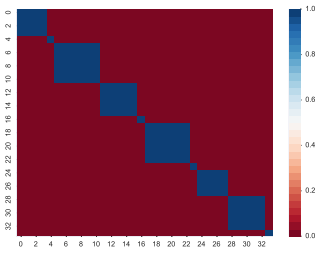


Figura 5.5. Mapa de calor de la matriz A para textos con 100 *frazes*

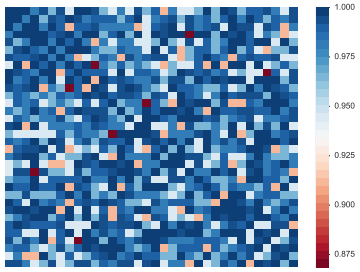
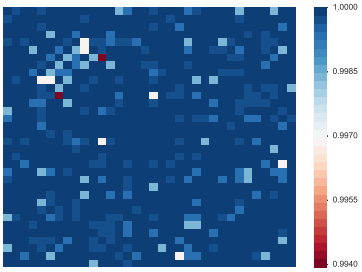
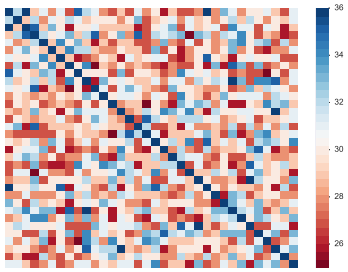
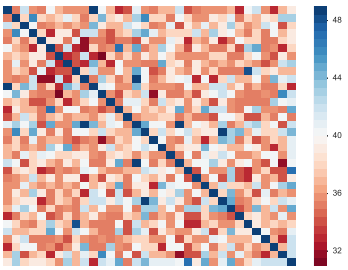
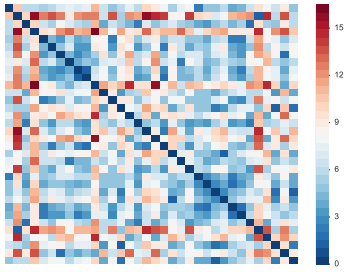
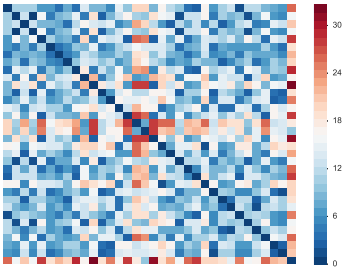
Métrica \ Clustering	SOMS	CBE
Similitud por probabilidad de número de coincidencias: $Z = (\zeta_{ij})$		
Similitud por número de coincidencias: $M = (m_{ij})$		
Similitud por proporciones: $H = (\eta_{ij})$		

Figura 5.6. Mapas de calor de las matrices de similitud para textos con 100 *frazes* codificados con CENG

5.1.3. Textos con 50 frazes

Las bases de datos se generaron de la misma forma que en los experimentos anteriores. Nuevamente, el único parámetro que varió fue que para este experimento se tomaron en cuenta textos que generaban bases de datos de al menos 50 *frazes*. Así, se analizaron 54 textos de 10 autores distintos.

I. Determinación del número de cúmulos

En las gráficas de la figura 5.7 se pueden ver los $WCSS$ para dos textos de la muestra:

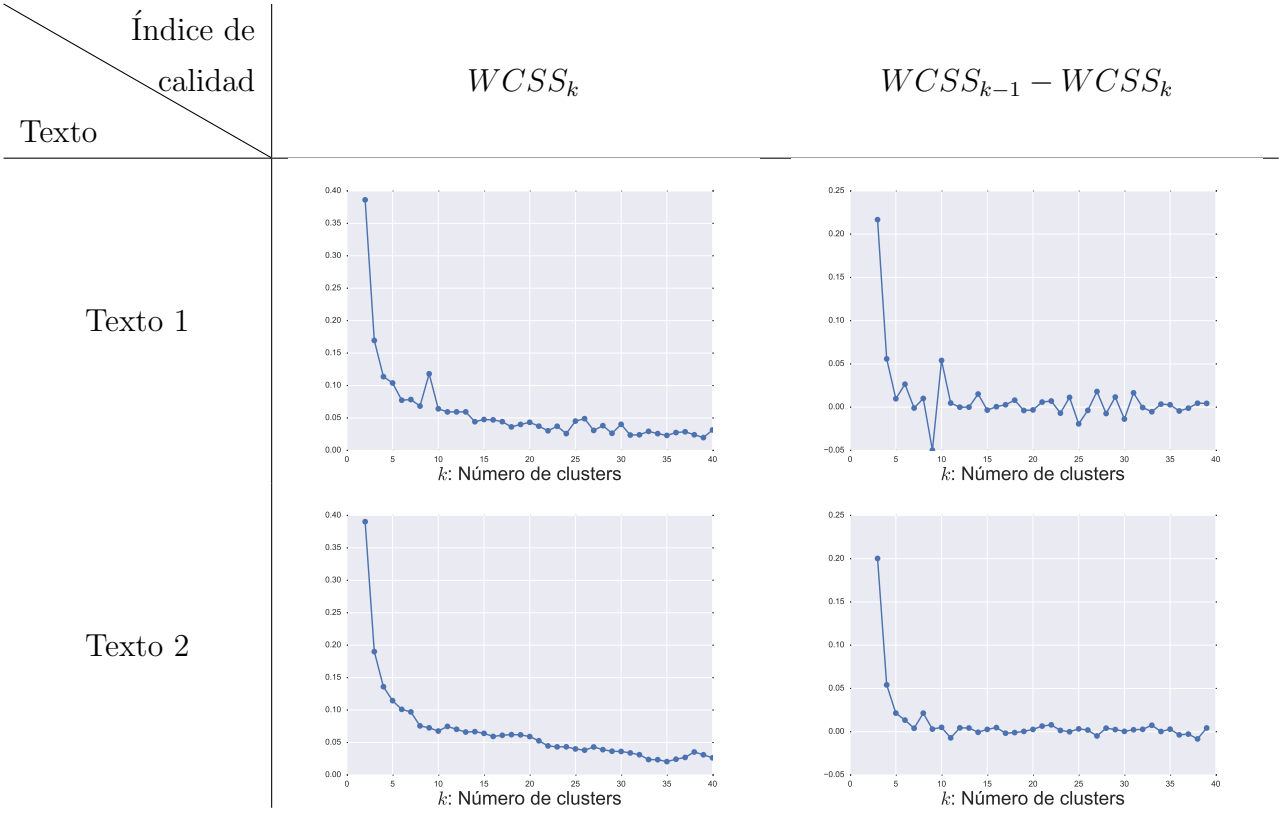


Figura 5.7. Gráficas del índice de calidad para textos con 50 *frazes* codificados con CENG

Esto permitió concluir que se utilizaría una $k = 6$ para un número de cúmulos.

II. Resultados de similitud

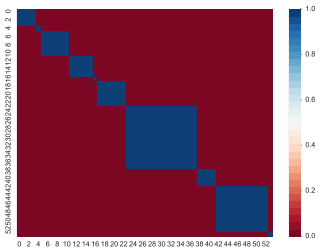


Figura 5.8. Mapa de calor de la matriz A para textos con 50 *frazes*

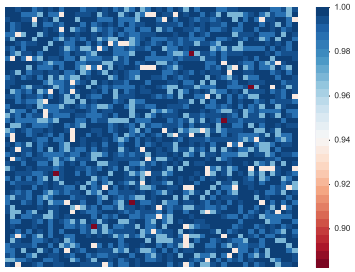
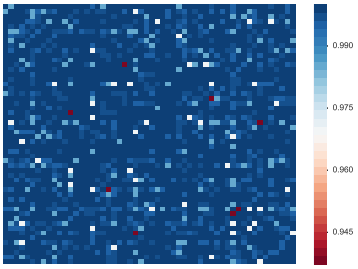
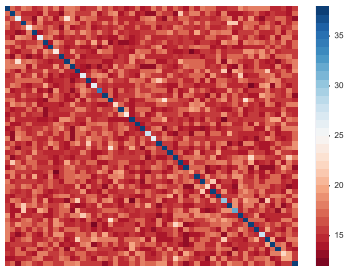
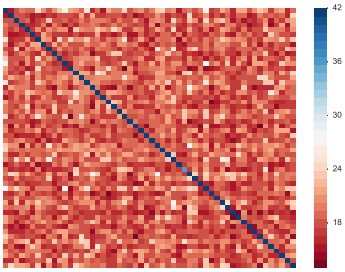
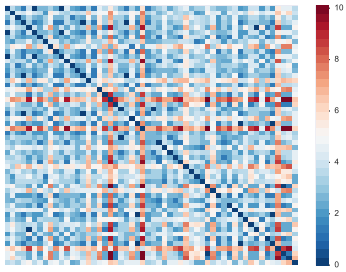
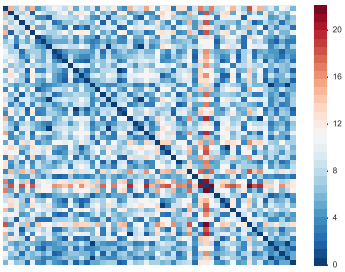
Métrica \ Clustering	SOMS	CBE
Similitud por probabilidad de número de coincidencias: $Z = (\zeta_{ij})$		
Similitud por número de coincidencias: $M = (m_{ij})$		
Similitud por proporciones: $H = (\eta_{ij})$		

Figura 5.9. Mapas de calor de las matrices de similitud para textos con 50 *frazes* codificados con CENG

5.1.4. Textos con 200 frazes tomados de [1]

A partir de los resultados anteriores que parecen no ser muy alentadores, se decidió poner a prueba el algoritmo de procesamiento numérico frente a una colección de datos utilizada exitosamente en el pasado. Esta muestra consta de 6 textos de 2 autores distintos. La estructura de la base de datos es distinta a la propuesta en el trabajo y se basa únicamente en la frecuencia de aparición de las palabras en el texto.

I. Determinación del número de cúmulos

En las gráficas de la figura 5.10 se pueden ver los $WCSS$ para dos textos de la muestra:

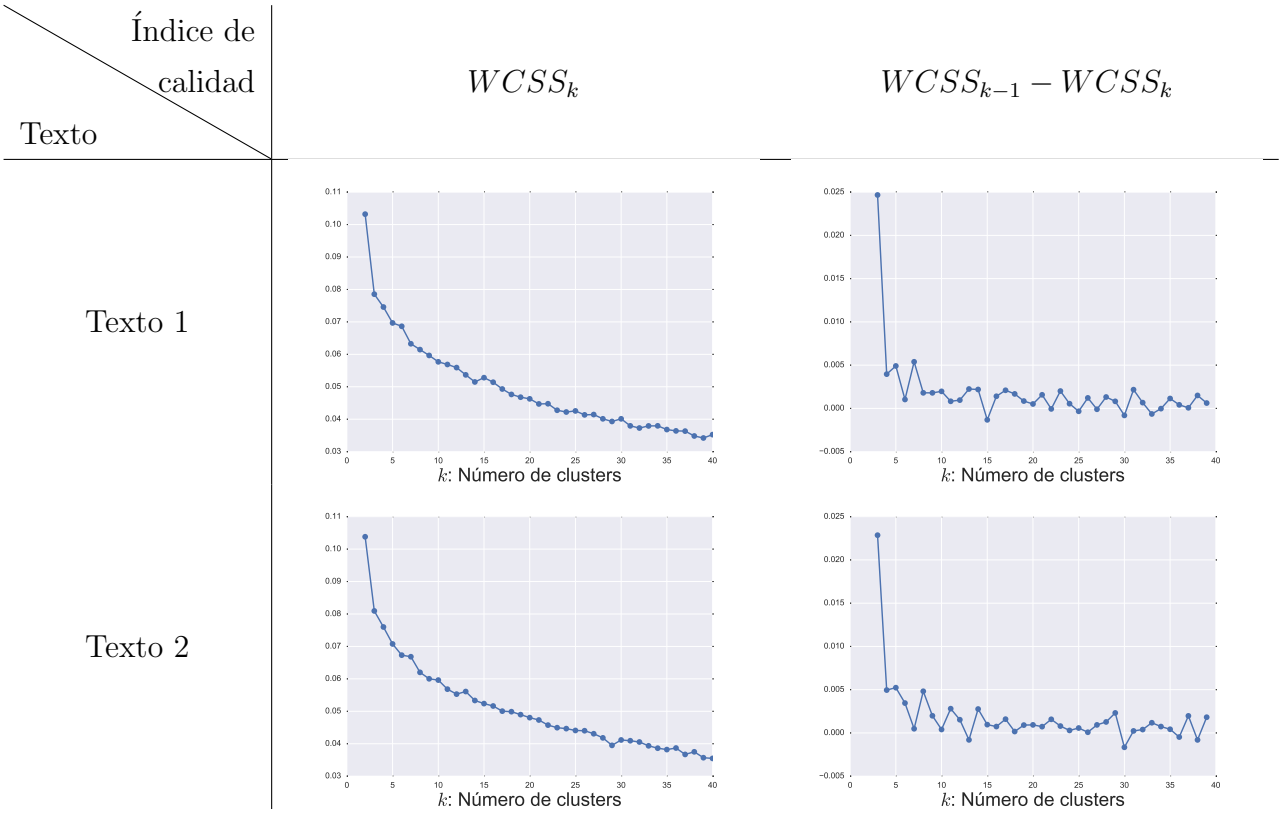


Figura 5.10. Gráficas del índice de calidad para textos con 200 *frazes* tomados de [1] codificados con CENG

Esto permitió concluir que se utilizaría una $k = 5$ para un número de cúmulos.

II. Resultados de similitud

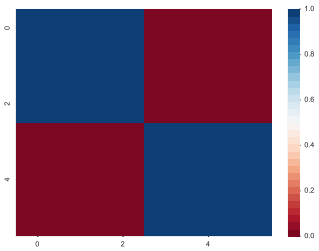


Figura 5.11. Mapa de calor de la matriz A para textos con 200 *frazes* tomados de [1]

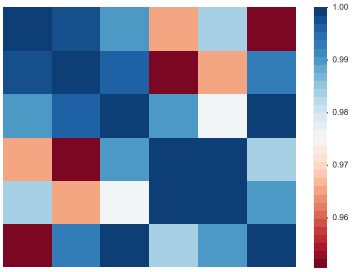
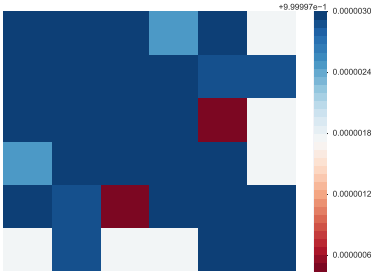
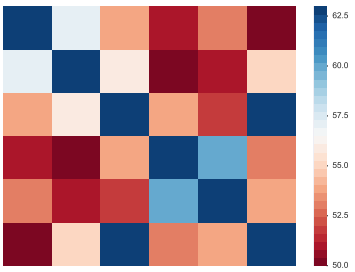
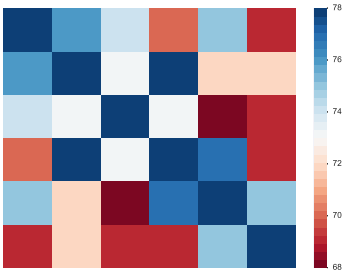
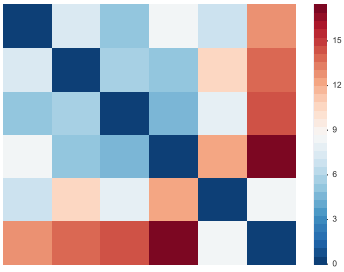
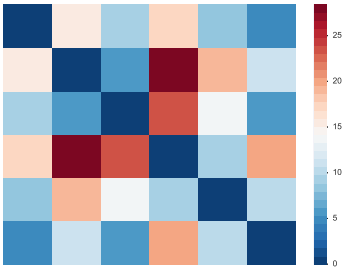
Métrica \ Clustering	SOMS	CBE
Similitud por probabilidad de número de coincidencias: $Z = (\zeta_{ij})$		
Similitud por número de coincidencias: $M = (m_{ij})$		
Similitud por proporciones: $H = (\eta_{ij})$		

Figura 5.12. Mapas de calor de las matrices de similitud para textos con 200 *frazes* tomados de [1] codificados con CENG

Ahora bien, si estos datos se presentaran en una tabla ordenada de acuerdo a la métrica de similitud, se pueden ver resultados un poco alentadores. De los seis textos analizados, tres pertenecen a Gabriel García Márquez y tres pertenecen a Julio Cortázar. Sin pérdida de generalidad, se etiqueta cada texto con dos iniciales del nombre de su autor, y un índice. En la parte izquierda de la tabla 5.1 se muestran los resultados reportados en la matriz M para el agrupamiento con mapas auto-organizados ordenados de mayor a menor para todas las posibles comparaciones entre parejas de textos. En la parte derecha de la tabla se muestran los resultados reportados en [1] ordenados de mayor a menor utilizando una métrica de similitud p_{ij} en cuyo detalle no se entrará. Las comparaciones entre textos del mismo autor se encuentran resaltadas.

Textos comparados	m_{ij}	Textos comparados en [1]	p_{ij}
GM3 JC3	63	**JC1 JC2**	98.23 %
JC1 JC2	60	**JC2 JC3**	86.43 %
GM1 GM2	57	**JC1 JC3**	82.50 %
GM2 GM3	56	**GM1 GM2**	78.54 %
GM2 JC3	55	**GM1 GM3**	74.69 %
JC2 JC3	54	GM3 JC3	68.80 %
GM1 GM3	54	**GM2 GM3**	66.77 %
GM3 JC1	54	GM1 JC3	65.71 %
JC1 JC3	53	GM2 JC1	57.14 %
GM1 JC2	53	GM3 JC2	57.14 %
GM3 JC2	52	GM2 JC3	54.29 %
GM1 JC1	51	GM1 JC2	54.29 %
GM2 JC2	51	GM3 JC1	54.29 %
GM1 JC3	50	GM1 JC1	51.51 %
GM2 JC1	50	GM2 JC2	48.57 %

Tabla 5.1. Comparativa con los resultados presentados en [1]

Se puede ver que en la parte inferior de la tabla se encuentran más parejas de textos de distintos autores, en ambos casos. En la parte superior, los resultados de [1] parecen indicar que salvo por una comparación (GM3 con JC3), se podría decir que la métrica

p_{ij} permite distinguir cuando se trata del mismo autor. Sin embargo, en los resultados del lado izquierdo, aparecen más comparaciones entre autores distintos evaluadas con un mayor valor para m_{ij}.

5.1.5. Textos con 500 frazes usando variables indicadoras booleanas

Con base en los resultados anteriores se optó por comparar el enfoque de codificación categórica del algoritmo CENG y manejar las variables categóricas utilizando un esquema de codificación tradicional. Para este experimento se utilizaron los mismos datos categóricos que en la sección 5.1.1 pero se codificaron con variables indicadoras booleanas incrementando el número de dimensiones de la base de datos. Posteriormente se realizó el mismo análisis numérico que se hizo para experimentos anteriores.

I. Determinación del número de cúmulos

En las gráficas de la figura 5.13 se pueden ver los $WCSS$ para algunos de los textos de la muestra:

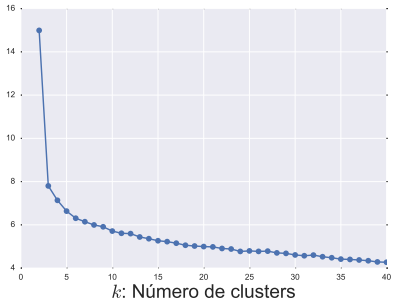
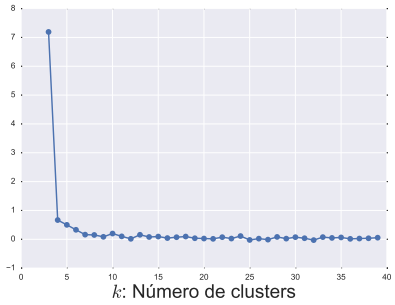
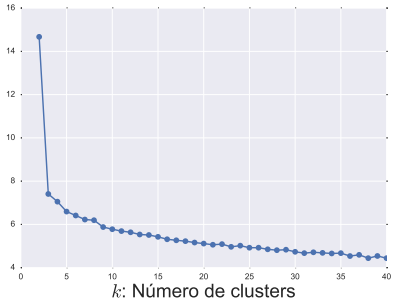
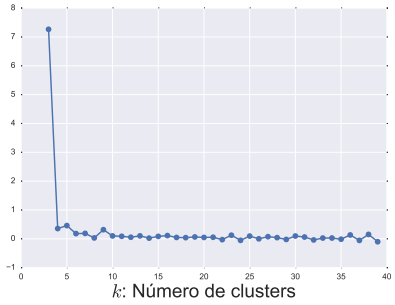
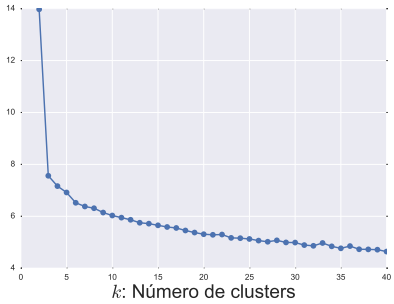
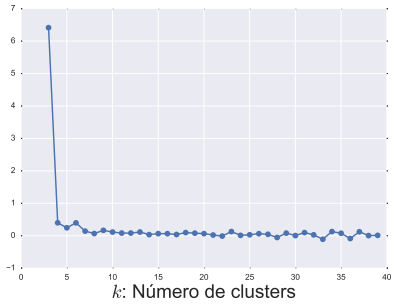
Índice de calidad / Texto	$WCSS_k$	$WCSS_{k-1} - WCSS_k$
Texto 1		
Texto 2		
Texto 3		

Figura 5.13. Gráficas del índice de calidad para textos con 500 *frazes* codificados con variables indicadoras booleanas

Esto permitió concluir que se utilizaría una $k = 6$ para un número de cúmulos.

II. Resultados de similitud

El mapa de calor ideal coincide con el de la figura 5.2.

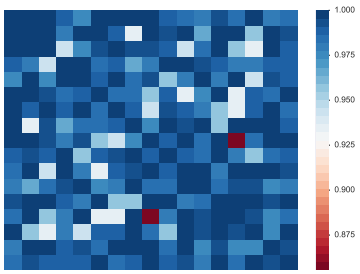
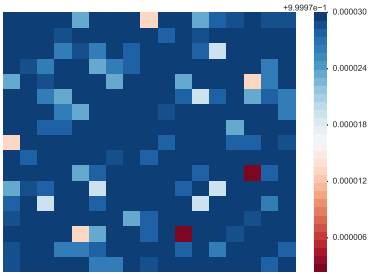
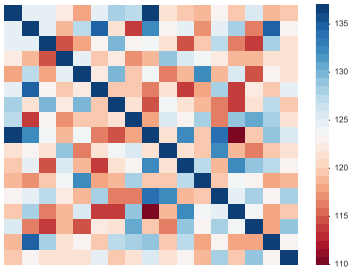
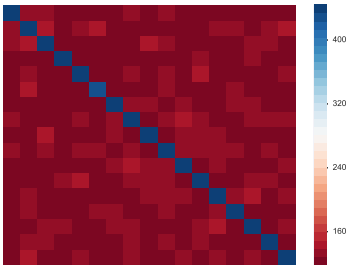
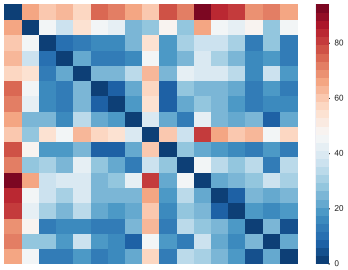
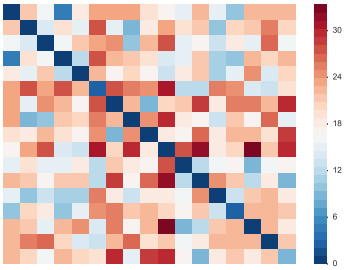
Métrica \ Clustering	SOMS	CBE
Similitud por probabilidad de número de coincidencias: $Z = (\zeta_{ij})$		
Similitud por número de coincidencias: $M = (m_{ij})$		
Similitud por proporciones: $H = (\eta_{ij})$		

Figura 5.14. Mapas de calor de las matrices de similitud para textos con 500 *frazes* codificados con variables indicadoras booleanas

5.1.6. Textos con 200 frazes tomados de [1] usando variables indicadoras booleanas

I. Determinación del número de cúmulos

En las gráficas de la figura 5.15 se pueden ver los $WCSS$ para algunos de los textos de la muestra:

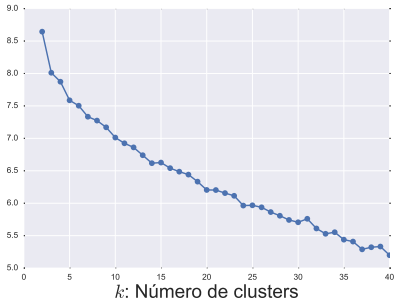
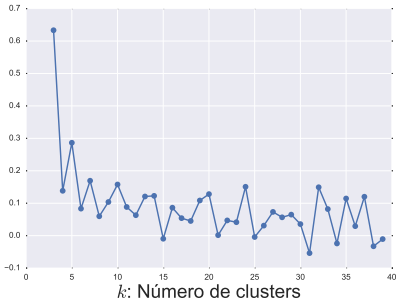
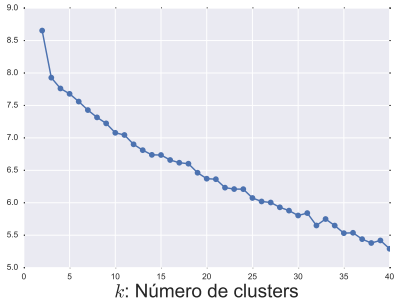
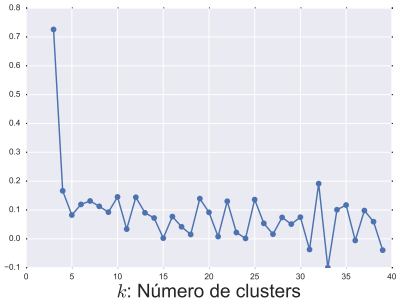
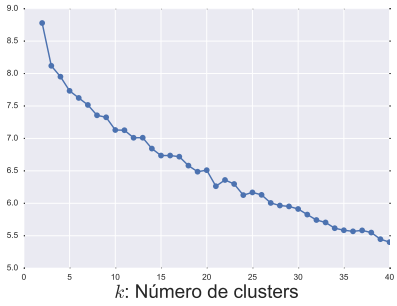
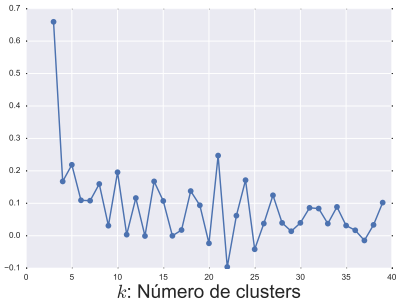
Índice de calidad / Texto	$WCSS_k$	$WCSS_{k-1} - WCSS_k$
Texto 1		
Texto 2		
Texto 3		

Figura 5.15. Gráficas del índice de calidad para textos con 200 *frazes* tomados de [1] codificados con variables indicadoras booleanas

Esto permitió concluir que se utilizaría una $k = 5$ para un número de cúmulos.

II. Resultados de similitud

El mapa de calor ideal coincide con el de la figura 5.11.

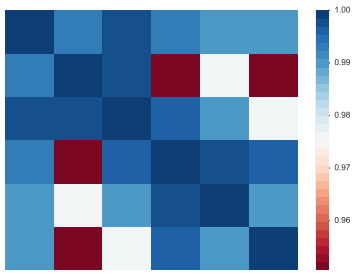
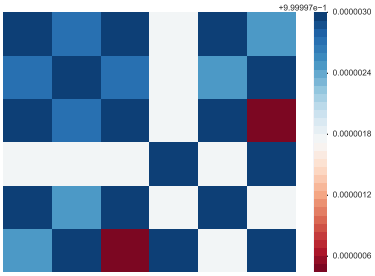
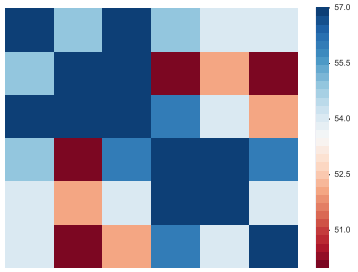
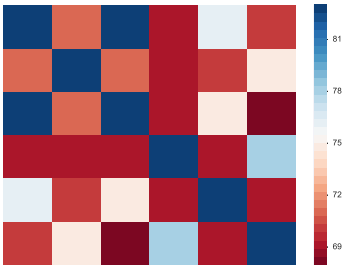
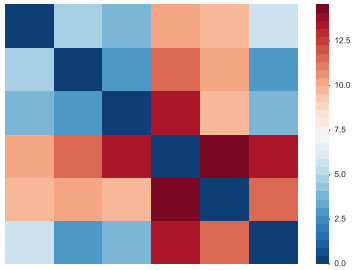
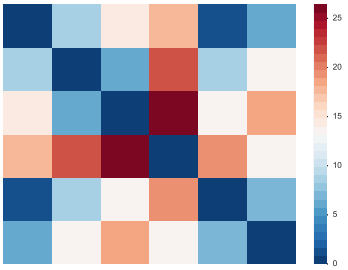
Clustering / Métrica	SOMS	CBE
Similitud por probabilidad de número de coincidencias: $Z = (\zeta_{ij})$		
Similitud por número de coincidencias: $M = (m_{ij})$		
Similitud por proporciones: $H = (\eta_{ij})$		

Figura 5.16. Mapas de calor para las matrices de similitud para textos con 200 *frazes* tomados de [1] codificados con variables indicadoras booleanas

La tabla 5.2 es análoga a la que se mostró en la 5.1, utilizando la matriz M y el algoritmo de agrupamiento basado en mapas auto-organizados. Nuevamente se resaltan las comparaciones entre los mismos autores. En este caso, se puede ver una distinción un poco más clara que la presentada en la sección 5.1.4.

Aunque las tablas 5.1 y 5.2 parecen presentar resultados más cercanos a los obtenidos en [1], el método presentado en este trabajo no parece indicar que se pueda diferenciar claramente entre autores. Más aún, la semejanza de resultados ilustrada en las tablas antes mencionadas, puede deberse simplemente al reducido tamaño de la muestra de textos tomada para estos ejemplos.

Textos comparados	m_{ij}	Textos comparados en [1]	p_{ij}
JC1 JC2	57	**JC1 JC2**	98.23 %
GM2 GM3	57	**JC2 JC3**	86.43 %
GM1 GM3	57	**JC1 JC3**	82.50 %
GM3 JC1	56	**GM1 GM2**	78.54 %
JC1 JC3	56	**GM1 GM3**	74.69 %
GM1 GM2	55	GM3 JC3	68.80 %
GM1 JC1	55	**GM2 GM3**	66.77 %
JC2 JC3	54	GM1 JC3	65.71 %
GM1 JC2	54	GM2 JC1	57.14 %
GM3 JC2	54	GM3 JC2	57.14 %
GM1 JC3	54	GM2 JC3	54.29 %
GM3 JC3	52	GM1 JC2	54.29 %
GM2 JC2	52	GM3 JC1	54.29 %
GM2 JC3	50	GM1 JC1	51.51 %
GM2 JC1	50	GM2 JC2	48.57 %

Tabla 5.2. Comparativa con los resultados presentados en [1]

5.2. Interpretación y análisis de resultados

Además de los experimentos aquí reportados, se corrieron algunas otras ligeras variaciones con resultados similares. Dada la extensión actual del trabajo y que no aportan mucha más información de la que ya se ha reportado, se omiten en este capítulo.

Sorprendentemente, se puede observar que en términos generales los mapas de calor de las matrices de similitud no se asemejan mucho a la estructura esperada ilustrada con la matriz A en ninguno de los casos. Esto indica que el enfoque propuesto en este trabajo quizás no es el adecuado para discernir entre autores distintos. A continuación se identifica con detalle cada uno de los puntos en los que el análisis podría estar fallando.

5.2.1. Análisis numérico de los datos

I. Algoritmos de agrupamiento y sus métricas

Los dos algoritmos de agrupamiento utilizados, utilizan métricas distintas y las optimizan de diferentes maneras. Los mapas auto-organizados están basados en distancia euclidiana y dependen de la topología de la red que se va a usar para encontrar los cúmulos. El agrupamiento entrópico minimiza la suma de varianzas de cada cúmulo en cada dimensión (utilizando distancia euclidiana) al tiempo que busca maximizar la entropía de la distribución del agrupamiento para no asumir nada que no esté implicado por los datos.

Ambos algoritmos son especialmente buenos para encontrar cúmulos de formas poco tradicionales (a diferencia de las que generaría por ejemplo una vecindad de radio 1 de un punto en $\mathbb{R}^n$ utilizando distancia euclidiana). Sin embargo, ambos fracasan al utilizarse para intentar distinguir entre autores. Esto podría sustentarse en el hecho de que ambos algoritmos tienen una componente importante basada en distancia euclidiana y quizás ese enfoque sea inapropiado para generar información a partir de una base de datos que aunque codificada numéricamente contiene información que es de carácter categórico.

Una vertiente que se podría explorar en trabajo futuro sería utilizar estos algoritmos de agrupamiento con otras métricas. Asimismo, se podrían poner a prueba otros algoritmos de agrupamiento que pudieran ser capaces de detectar algún patrón en los datos que permita discernir entre autores.

II. Mediciones de similitud

Otro punto en el que el análisis numérico de los datos podría no estar siendo del todo preciso es la forma en la que se está midiendo similitud. Las métricas de similitud que se presentaron en la sección 4.7 están construidas de forma completamente intuitiva, buscando medir la similitud entre dos agrupamientos resultantes.

Esto es una pregunta interesante que ha sido abordada por autores como Reilly et.al. en un trabajo [52] en donde se propone un enfoque similar al aquí planteado. Sobre esta línea se podrían explorar otras formas de comparar agrupamientos que resulten más efectivas.

5.2.2. Algoritmo de codificación categórica

El objetivo principal de este trabajo era utilizar exitosamente la codificación numérica de salida del CENG para una base de datos con variables categóricas para hacer análisis de autoría sobre textos. Otro punto en el que la solución propuesta podría estar fallando sería en la codificación que arroja el CENG.

Se podría también profundizar en el análisis de la parametrización del CENG expuesto en la sección 4.4.3. Variar los parámetros del CENG podría generar una codificación que resulte más efectiva para detectar las diferencias entre estilos de escritura de diferentes autores.

Al concluir los experimentos expuestos en las secciones 5.1.1, 5.1.2, 5.1.3 y 5.1.4 y obtener resultados no muy favorables, se optó por poner a prueba la codificación del CENG y repetir el análisis para las mismas bases de datos pero codificando ahora las variables categóricas utilizando variables indicadoras booleanas. Dicho tratamiento de las variables categóricas es comúnmente utilizado y aunque incrementa el número de dimensiones de la base de datos, resulta efectivo.

Como se puede ver en los resultados de los experimentos 5.1.5 y 5.1.6, parece tampoco haber ninguna mejora significativa al utilizar variables indicadoras booleanas. Todo esto parece indicar que el esquema de codificación no es el problema principal.

5.2.3. Estructura de la base de datos

Por último, una posibilidad muy importante que podría explicar todos los resultados obtenidos en estos experimentos es que la estructura de la base de datos no esté bien definida. Todos los posibles puntos de falla mencionados anteriormente podrían estar funcionando a la perfección, pero si están operando sobre datos que no contienen los patrones que se busca identificar, el análisis se está haciendo en vano.

La estructura de *frazes* que se definió en la sección 4.5 tiene asociada una pérdida de información bastante alta. Al utilizar un modelo tan simple como una *fraze* para representar un texto no-estructurado se puede estar perdiendo la información que podría permitir distinguir entre el estilo de dos autores.

Si se analizan con detalle los mapas de calor de las matrices Z que miden la probabilidad de que dos agrupamientos coincidan en cierto número de posiciones se puede observar

que para la mayoría de los casos estos valores son muy elevados. Esto implica que las probabilidades de que dos agrupamientos coincidieran en ese número de posiciones eran muy bajas. Una posible interpretación de esto es que los agrupamientos generados eran en efecto muy similares; tanto que eran prácticamente indistinguibles. Esto último puede ser a causa de que la estructura de las *frazes* en efecto no modela el estilo de escritura del autor si no alguna característica más general que tienen en común todos los textos analizados.

Aunque la estructura de la base de datos se definió cuidadosamente y está basada en marcadores estilísticos que han sido utilizados con éxito en el pasado para realizar análisis de autoría puede resultar muy limitada. Por ejemplo, en la sección 2.2 se menciona el artículo de Chen [11] que utiliza 120 marcadores estilísticos distintos. En este trabajo se tuvo que reducir la complejidad de la estructura de la base de datos para adaptarnos a restricciones en el tiempo de procesamiento del algoritmo. Otra línea sobre la que se podría trabajar en el futuro sería definir estructuras más complejas que busquen minimizar la pérdida de información entre el texto original y la base de datos estructurada.

6. Conclusiones

If you can't explain it simply, you don't understand it well enough.

– Albert Einstein

En este capítulo se presenta un breve resumen del trabajo expuesto en esta tesis. Se comentan brevemente los resultados obtenidos y se indican posibles áreas de trabajo futuro.

6.1. Resumen

En este trabajo se buscó programar un sistema que permitiera identificar si dos textos habían sido escritos por el mismo autor. Para ello, se identificaron un conjunto de caracterísitcas lingüísticas que han sido utilizadas en trabajos anteriores para atribuir autoría de textos. Con base en dichas caracterísitcas, se definió la estructura de una base de datos con variables tanto categóricas como numéricas que se utilizaría para modelar los textos a analizar.

Se elaboró un sistema de pre-procesamiento que identifica las categorías gramaticales de las palabras de los textos de entrada (POS-tags) usando un etiquetador pre-entrenado que se distribuye como parte de Stanford CoreNLP. Se programó también un sistema que llena la estructura definida anteriormente con las categorías gramaticales presentes en el texto etiquetado. Las bases de datos resultantes se le alimentaron al algoritmo de codificación categórica CENG, para obtener a la salida bases de datos puramente numéricas. El CENG se parametrizó con precisión para obtener un buen desempeño con un tiempo de cómputo razonable. Finalmente, se definió un algoritmo de procesamiento numérico para poder medir la similitud entre dos bases de datos codificadas. Dicho algoritmo se diseñó de forma general para poder ser aplicado a cualquier conjunto de

bases de datos numéricas resultantes del CENG, sin importar si modelan texto, música, pintura o cualquier otra cosa. Para poder aplicar el enfoque propuesto en esta tesis a otras disciplinas, bastaría con definir la forma de pre-procesar los datos (la música, por ejemplo) para modelarla como una base de datos con variables categóricas.

El algoritmo numérico busca medir la similitud entre dos bases de datos comparando el patrón de cúmulos que resulta de correr el mismo algoritmo de agrupamiento sobre ambas bases de datos. Para ello se probaron dos algoritmos de agrupamiento: mapas auto-organizados y un agrupamiento basado en entropía. Se identificó el número de cúmulos utilizando el método del codo y se alimentó este parámetro a los algoritmos de agrupamiento. Los agrupamientos resultantes se compararon entre sí utilizando tres métricas de similitud construídas para comparar patrones de agrupamiento.

Finalmente, se evaluó el sistema completo en tres conjuntos de experimentos. El primero evalúa la capacidad del sistema para discernir entre autores sobre tres conjuntos de modelos de textos generados a partir de un corpus original y utilizando la estructura definida en este trabajo. El segundo hacía lo mismo pero para un conjunto de datos utilizado en trabajo previo [1] en el cual la estructura de la base de datos está definida de una forma mucho más sencilla que en este trabajo, y no involucra tantas consideraciones *a priori*. Por último, se evaluó uno de los conjuntos de datos de los definidos en este trabajo y el conjunto de datos utilizado en [1] cambiando el esquema de codificación de variables categóricas por una codificación utilizando variables indicadoras booleanas. En todos los casos, el algoritmo de procesamiento numérico utilizado fue el mismo.

En la sección 5.2 se comentaron y se analizaron los resultados obtenidos en los experimentos realizados. Dado que los resultados no fueron favorables, en esa sección se comentan todos los puntos en los que podría mejorarse el sistema.

Cabe mencionar que la motivación de este trabajo, es un artículo [1] en el que se ilustra un experimento significativamente más sencillo que el aquí presentado. Las bases de datos de dicho enfoque, están basadas puramente en la frecuencia de aparición de palabras en los textos. Introducir rasgos léxicos en el pre-procesamiento del texto busca mejorar los resultados al enriquecer la información que se le está presentando al CENG. Sin embargo, es posible que esta información adicional no solo no sea útil para distinguir entre estilos de escritura distintos, si no que introduzca atractores indeseados que sesguen el proceso de búsqueda por los mejores códigos.

6.2. Trabajo futuro

En la sección 5.2 ya se mencionaron algunas de las áreas en las que se puede realizar trabajo futuro para mejorar el desempeño de este algoritmo y poder distinguir entre autores satisfactoriamente.

Puntualmente, este trabajo se puede ampliar trabajando en lo siguiente:

- Definir una estrucutra más compleja para la base de datos, que tome en cuenta un mayor número de marcadores estilísticos a costa de un mayor tiempo de procesamiento para obtener la codificación de las variables con el CENG.
- Analizar a profundidad la parametrización y los códigos arrojados por el CENG. Nuevamente, a costa de un mayor tiempo de procesamiento, se podrían obtener códigos que mejoren el desempeño del algoritmo.
- Modificar el algoritmo de procesamiento numérico para comparar bases de datos numéricas explorando tres vertientes:
  - Trabajar con otros algoritmos de agrupamiento y cambiar las métricas que dichos algoritmos optimizan por otras que permitan detectar otro tipo de patrones. Por ejemplo, se podría buscar minimizar la entropía intra-cúmulo en lugar de la varianza.
  - Refinar las métricas definidas para medir similitud entre agrupamientos o definir nuevas métricas.
  - Idear un algoritmo de procesamiento numérico completamente nuevo que a partir de dos bases de datos numéricas, sea capaz de decir qué tan similares son.
- Una vez que funcione el sistema para distinguir entre autores, se podría entrenar un modelo que identifique los patrones que comúnmente se encuentren en textos de un autor y con base en eso sea capaz de clasificar un texto nuevo como escrito por alguno de los autores que ya se conocen.
- Suponiendo que el CENG esté bien parametrizado y el algoritmo numérico permita medir similitud entre agrupamientos de forma efectiva, se puede experimentar con datos que no sean texto; por ejemplo música. Para ello sería necesario llevar a cabo un estudio de los rasgos que permitan distinguir entre estilos en esta discipli-

na. Con base en esos rasgos habría que definir cómo procesar los datos de entrada para llevarlos al formato aceptado por el CENG: una base de datos estructurada que conste de al menos una variable categórica.

Apéndice A: Reducción de instancias categóricas

En este apéndice se presenta una alternativa para reducir aún más el número de instancias categóricas en las bases de datos que se le alimentan al CENG. Tras analizar las consecuencias que podría tener dicha reducción y el ligero impacto en el tiempo de ejecución al no hacerla, se optó por no incluirla en los experimentos realizados. Sin embargo, se considera importante explicar una manera en la que se podrían reducir el número de instancias, de ser necesario.

Con base en las frecuencias reportadas en las tablas 4.2 y 4.3 y a los resultados obtenidos en la sección 4.4.2 originalmente se planteó una reducción del número de instancias categóricas basada en la frecuencia de aparición de cada etiqueta. Por ejemplo, para el caso de los pronombres, existen 9 instancias. Para reducir ese número a sólo 3 se optó por crear 3 "meta-instancias" de tal forma que la primera meta-instancia contenga suficientes instancias para cubrir aproximadamente el primer 33.3 % de la frecuencia relativa de las instancias elegidas al total de palabras de esa categoría. La segunda meta-instancia contendría el segundo tercio de frecuencia relativa y así sucesivamente. Esta reducción queda mejor ilustrada en la figura A.1.

Instancia	**Frecuencia absoluta**	**Frecuencia relativa**
pp000000	6,115	0.3329
pr000000	5,464	0.2974
p0000000	3,453	0.1880
pi000000	1,842	0.1003
pd000000	647	0.0352
pt000000	607	0.0330
pn000000	182	0.0099
px000000	53	0.0029
pe000000	7	0.0004

$\Longrightarrow$

Etiquetas incluídas	**Frecuencia relativa**	**Meta-instancia**
pp000000	0.3329	P0
pr000000	0.4854	P1
p0000000		
pi000000	0.1817	P2
pd000000		
pt000000		
pn000000		
px000000		
pe000000		

Figura A.1. Reducción de instancias categóricas a partir de meta-instancias

En la tabla A.1 se pueden ver las categorías gramaticales que se hubieran tomado en cuenta para el análisis del texto en cada variable categórica de la base de datos, cuántas variables de ese tipo habrá y cuántas instancias existirán en cada una de ellas. Esta tabla se puede comparar con la tabla 4.5 que ilustra el número de instancias que en realidad se utilizaron.

Primera letra del tag	**Número de categorías**	**Número de instancias**
A	1	2
C	1	2
D	1	4
F	1	4
N	2	5
P	1	3
R,S	1	3
V	1	6
I,W,Z	1	3
3	0	0
Total	10	32

Tabla A.1. Elección de categorías y número de instancias para el modelo de oración

La justificación principal para ya no llevar a cabo el análisis de esta forma tiene que ver con la granularidad con la que está representada la información. Si se reduce el número de instancias categóricas con el enfoque aquí propuesto, se obtiene menos información a partir del texto. Lo que sí es importante resaltar, es que existe un importante desbalance entre las cantidad de etiquetas distintas que están definidas en los estándares EAGLES [46] para cada categoría gramatica. Esto hace que ciertas categorías gramaticales provean información más rica que otras.

Apéndice B: Fragmentos de código

En este apéndice se presentan algunos fragmentos del código escrito para llevar a cabo tanto el pre-procesamiento del texto como el procesamiento numérico para medir similitud. Está escrito en Python 3.4 y con la instalación de los paquetes indicados se puede ejecutar sin problemas. El código de Python completo se puede consultar en: `https://www.github.com/jvrsgsty/itam-tesis`

Ciclo para el llenado de las bases de datos

El siguiente fragmento de código muestra la forma en la que se llena una base de datos con *frazes*, una vez que se ha definido su estructura y el criterio de corte entre *fraze* y *fraze*. En términos generales es un ciclo que itera sobre todas las palabras presentes en un texto, una vez que han sido etiquetadas con su categoría gramatical. La *fraze* se define como un diccionario vacío, en la que todas las variables categóricas inician con un valor `NA` y las variables numéricas con 0.

Se itera mientras que no se haya llenado la *fraze* y no se hayan acabado las palabras del texto. Para cada palabra, se obtiene su categoría y se intenta colocar dentro de la *fraze* actual. Si ya se ha llenado esa posición de la *fraze* se descarta esa palabra; si no, se coloca. Se incrementan los contadores de las variables numéricas. En cuanto la *fraze* está llena (lo indica el método auxiliar `isFull`), se guardan los valores de las variables numéricas, se convierten los valores nulos (`NA`) en valores distintos por columna y se agrega la *fraze* a la base de datos.

El proceso se repite con una nueva *fraze* hasta que se terminen las palabras del texto.

```
while i < len(all_words):
  zentence = {'A': 'NA', 'C': 'NA', 'D': 'NA', 'F': 'NA', 'N_1': 'NA',
              'N_2': 'NA', 'P': 'NA', 'RS': 'NA', 'V': 'NA', 'IWZ': 'NA',
              '_tokens': 0, '_words': 0, '_punct': 0}
  num_tokens = 0
  num_words = 0
  num_punctuation = 0
  keys = sorted(list(zentence.keys()))
  # Fill a zentence up
  while not isFull(zentence, num_tokens) and i < len(all_words):
    w = all_words[i]
    tag = w.split('_')[1].rstrip()
    cat = tag[0].upper()
    j = 0
    l = len(keys) - NUM_NUMERICAL
    is_filled = False
    # For a given word, iterate through the whole zentence and try to make it fit
    while not is_filled:
      word_cats = keys[j].split("_")[0]
      if cat in word_cats and zentence[keys[j]] == 'NA':
        zentence[keys[j]] = tag
        is_filled = True
      j += 1
    if isWord(tag):
      num_words += 1
    if isPunctuation(tag):
      num_punctuation += 1
    num_tokens += 1
    i += 1
  # Once a zentence is full, add the numeric values
  if isFull(zentence, num_tokens):
    zentence['_tokens'] = num_tokens
    zentence['_words'] = num_words
    zentence['_punct'] = num_punctuation
    zentence = uniquifyNAs(zentence) # Append category name to NA values
    database += [zentence]
```

Métodos auxiliares para el llenado de las bases de datos

El siguiente fragmento de código simplemente contiene los métodos auxiliares a los que se hizo referencia en el ciclo anterior. El método `isFull` revisa si la *fraze* ya está llena de acuerdo a los criterios definidos en la sección 4.5. El método `isWord` revisa si la etiqueta dada corresponde a una palabra. El método `isPunctuation` revisa si la etiqueta dada corresponde a un signo de puntuación. El método `uniquifyNAs` hará que los `NA`s que queden al final en cada *fraze* sean tratados como instancias distintas si se encuentran en columnas distintas.

```
from collections import Counter
NUM_NUMERICAL = 3    # Number of numerical variables in the zentence structure
NUM_TOKENS = 12      # Max number of tokens to analyze in each zentence
PERCENT_NA = 0.3     # Max percentage of NAs admissible in a zentence

def isFull(zentence, num_tokens):
  """Determines whether or not a given zentence can be considered full
  Args:
    zentence(dict): a zentence dictionary where values not yet filled should
     be equal to 'NA'
    num_tokens(int): How many tokens have been analyzed to fill the current
      zentence up
  """
  c = Counter(zentence.values())
  is_full = c['NA']/(len(zentence) - NUM_NUMERICAL) <= PERCENT_NA
  is_full = is_full or num_tokens > NUM_TOKENS
  return is_full

def isWord(tag):
  return tag[0].upper() not in ['F', 'Z', 'W', '3']

def isPunctuation(tag):
  return tag[0].upper() == 'F'

def uniquifyNAs(zentence):
  for k,v in zentence.items():
    if v == 'NA':
      zentence[k] += '_' + k
  return zentence
```

Agrupamiento con Python y R

El siguiente fragmento de código muestra cómo se lleva a cabo el agrupamiento con mapas auto-organizados. Es interesante, pues Python se encarga de la lectura de los datos y a través del paquete `rpy2` se los envía a R para que ejecute el algoritmo de agrupamiento y entregue los resultados a Python. Las ventajas principales de este enfoque son que la sintaxis del código necesario para ejecutar el código de R es la de Python y que no se tiene que gestionar la comunicación entre dos procesos independientes, si no que desde el mismo programa de Python, se ejecuta el código de R como un subproceso.

El parámetro `data` simplemente contiene una matriz de datos en un `ndarray` y el parámetro `k` contiene un entero con el número de cúmulos que se buscarán con el algoritmo de agrupamiento. Basta con convertir los datos a un `data frame` de R y utilizar los objetos de Python que mapean a los objetos de R para ejecutar el algoritmo de agrupamiento.

El método regresa simplemente un vector de asignación como el de la definición 4.1.

```
1  import numpy as np
2  from rpy2.robjects import FloatVector, r
3  from rpy2.robjects.packages import importr
4  from rpy2.robjects.numpy2ri import numpy2ri
5  kohonen = importr('kohonen')
6
7  def calculateCenters(data, k):
8    data_train_matrix = numpy2ri(data)
9    grid = kohonen.somgrid(xdim=k, ydim=1, topo="rectangular")
10   kwargs = {'grid': grid,
11             'rlen': 500,
12             'radius': FloatVector([4.0, 0.3262874458]),
13             'alpha': FloatVector([0.999, 0.0065639126]),
14             'mode': 'batch',
15             'dist.fcts': 'euclidean'}
16   results = kohonen.som(data_train_matrix, **kwargs)
17   centers = np.array(results.rx2('codes')[0])
18   return centers
```

Mediciones de similitud y búsqueda de la permutación adecuada

El siguiente fragmento de código muestra la forma en la que se puede encontrar la permutación de etiquetas que hace que mejor coincidan dos agrupamientos distintos, como se explicó en la sección 4.7. Dados dos vectores de asignación (`m1` y `m2`) resultantes de ejecutar el mismo algoritmo de agrupamiento a dos bases de datos distintas, se permutan las etiquetas un máximo de 1000 veces y se busca el número de coincidencias máximo y la norma mínima para la diferencia de vectores de frecuencias de asignación (definición 4.6).

Regresa unos diccionarios anidados, con los valores que se requerirán para construir las tres diferentes matrices de similitud.

```python
import math, numpy as np

def findBestPermutation(k, m1, m2):
  best_idx_p = 0
  best_idx_n = 0
  min_non_zero = len(m1)
  min_norm = np.linalg.norm(np.ones(len(m1))*len(m1))
  # Preliminary variables for frequencies approach
  y = np.bincount(m1)
  ii = np.nonzero(y)[0]
  freq1 = y[ii]
  y = np.bincount(m2)
  ii = np.nonzero(y)[0]
  freq2 = y[ii]
  # randomly pick at most 1000 permutations of the indices
  if k >= 7: # 7! = 5040
    n = 1000
  else:
    n = math.factorial(k)
  for i in range(n):
    perm = np.random.permutation(k)
    # Coincidence probability approach
    vf = np.vectorize(lambda x: perm[x])
    mp = vf(m2)
    non_zero = np.count_nonzero(m1 - mp)
    if non_zero < min_non_zero:
      min_non_zero = non_zero
      best_perm_p = perm
    # Assignment ratio approach (frequencies)
    freqp = freq2[perm]
    norm = np.linalg.norm(freq1 - freqp)
    if norm < min_norm:
      min_norm = norm
      best_perm_n = perm
  coincidences = len(m1) - min_non_zero
  probability = {'coincidences': coincidences, 'perm': best_perm_p}
  frequencies = {'norm': min_norm, 'perm': best_perm_n}
  best = {'probability': probability, 'frequencies': frequencies}
  return best
```

Cálculo de la similitud por probabilidad de número de coincidencias

En el siguiente fragmento de código (en el método `coincidenceProbability`) se calcula la **similitud por probabilidad de número de coincidencias** entre el texto i y el texto j (ζ_{ij}), como aparece en la definición 4.4. En el código, `m` equivale a la m_{ij} de la definición, `n` equivale a T y `k` equivale a k.

Adicionalmente, se muestra cómo dados dos vectores de asignación `m1` y `m2`, primero se encuentra la mejor permutación utilizando el código de la sección anterior y luego se calcula ζ_{ij}.

Este fragmento regresa un diccionario con los valores de m_{ij}, ζ_{ij} y η_{ij} para la comparación entre dos textos. Si se ejecuta este código para todas las posibles parejas de textos de la muestra, se pueden construir las matrices de similitud definidas en la sección 5.1.

```
import numpy as np
from scipy import misc

def coincidenceProbability(m, k, n):
  s = 0
  for i in range(m, n+1):
    s += misc.comb(n, i, exact = True)*(k-1)**(n-i)
  return 1 - (s / k**n)

def similarity(m1, m2, k):
  n = len(m1)
  best = findBestPermutation(k, m1, m2)
  # Coincidence probability
  m = best['probability']['coincidences']
  s = coincidenceProbability(m, k, n)
  # Frequencies
  norm = best['frequencies']['norm']
  return {'coincidences': m, 'probability': s, 'frequencies': norm}
```

Referencias

[1] A. F. Kuri-Morales, "Mining unstructured data via computational intelligence," in *Advances in Artificial Intelligence and Soft Computing*. Springer, 2015, pp. 518–529.

[2] S. Haykin, *Neural Networks: A Comprehensive Foundation*, 2nd ed. Upper Saddle River, NJ, USA: Prentice Hall PTR, 1998.

[3] A. F. Kuri-Morales, "The best neural network architecture," in *Nature-Inspired Computation and Machine Learning*. Springer, 2014, pp. 72–84.

[4] M. L. Brocardo, I. Traore, S. Saad, and I. Woungang, "Authorship verification for short messages using stylometry," in *Computer, Information and Telecommunication Systems (CITS), 2013 International Conference on*. IEEE, 2013, pp. 1–6.

[5] G. Ledger and T. Merriam, "Shakespeare, fletcher, and the two noble kinsmen," *Literary and Linguistic Computing*, vol. 9, no. 3, pp. 235–248, 1994.

[6] D. I. Holmes and R. S. Forsyth, "The federalist revisited: New directions in authorship attribution," *Literary and Linguistic Computing*, vol. 10, no. 2, pp. 111–127, 1995.

[7] R. D. Peng and N. W. Hengartner, "Quantitative analysis of literary styles," *The American Statistician*, vol. 56, no. 3, pp. 175–185, 2002.

[8] A. Abbasi and H. Chen, "Writeprints: A stylometric approach to identity-level identification and similarity detection in cyberspace," *ACM Transactions on Information Systems (TOIS)*, vol. 26, no. 2, p. 7, 2008.

[9] F. Iqbal, R. Hadjidj, B. C. Fung, and M. Debbabi, "A novel approach of mining write-prints for authorship attribution in e-mail forensics," *digital investigation*, vol. 5, pp. S42–S51, 2008.

[10] F. Iqbal, H. Binsalleeh, B. C. Fung, and M. Debbabi, "Mining writeprints from anonymous e-mails for forensic investigation," *digital investigation*, vol. 7, no. 1, pp. 56–64, 2010.

[11] X. Chen, P. Hao, R. Chandramouli, and K. Subbalakshmi, "Authorship similarity detection from email messages," in *Machine Learning and Data Mining in Pattern Recognition*. Springer, 2011, pp. 375–386.

[12] O. De Vel, "Mining e-mail authorship," in *Proc. Workshop on Text Mining, ACM International Conference on Knowledge Discovery and Data Mining (KDD'2000)*, 2000.

[13] F. Esponda, "Everything that is not important: Negative databases [research frontier]," *IEEE Computational Intelligence Magazine*, vol. 3, no. 2, pp. 60–63, 2008.

[14] T.-Y. Qian, B. Liu, Q. Li, and J. Si, "Review authorship attribution in a similarity space," *Journal of Computer Science and Technology*, vol. 30, no. 1, pp. 200–213, 2015.

[15] S. Ferilli, "A sentence structure-based approach to unsupervised author identification," *Journal of Intelligent Information Systems*, vol. 46, no. 1, pp. 1–19, 2014. [Online]. Available: http://dx.doi.org/10.1007/s10844-014-0349-9

[16] J. Nocedal and S. Wright, *Numerical optimization*. Springer Science & Business Media, 2006.

[17] A. Ibarra Chaoul, "EGA, CMA y SA, Una competencia a la muerte. Análisis comparativo entre tres metaheurísticas." Tesis, Instituto Tecnológico Autónomo de México, Río Hondo No.1, Col. Progreso Tizapán, 01080 Ciudad de México, México, 2013.

[18] K. Sörensen, "Metaheuristics—the metaphor exposed," *International Transactions in Operational Research*, vol. 22, no. 1, pp. 3–18, 2015.

[19] A. F. Kuri-Morales, "Categorical encoding with neural networks and genetic algorithms," in *AICT 2015 (Applied Informatics and Computing Theory), Salerno, Italy*. WSEAS, 2015.

[20] J. H. Holland, *Adaptation in natural and artificial systems: an introductory analysis with applications to biology, control, and artificial intelligence.* U. Michigan Press, 1975.

[21] A. F. Kuri-Morales and E. Aldana-Bobadilla, "The best genetic algorithm i," in *Advances in Soft Computing and Its Applications*. Springer, 2013, pp. 1–15.

[22] A. F. Kuri-Morales, E. Aldana-Bobadilla, and I. López-Peña, "The best genetic algorithm ii," in *Advances in Soft Computing and Its Applications*. Springer, 2013, pp. 16–29.

[23] A. E. Eiben, E. H. Aarts, and K. M. Van Hee, "Global convergence of genetic algorithms: A markov chain analysis," in *Parallel problem solving from nature*. Springer, 1990, pp. 3–12.

[24] T. E. Davis and J. C. Principe, "A markov chain framework for the simple genetic algorithm," *Evolutionary computation*, vol. 1, no. 3, pp. 269–288, 1993.

[25] G. Rudolph, "Convergence analysis of canonical genetic algorithms," *Neural Networks, IEEE Transactions on*, vol. 5, no. 1, pp. 96–101, 1994.

[26] J. I. López Peña, "Planteamiento y avances del proyecto de investigación determining the position of a mobile robot by means of the spectral signature of images," http://jvrsgsty.github.io/itam/tesis/references/IgnacioLopez_DeterminingRobotPosition_2017.pdf, accesado: 2017-05-24.

[27] E. Zitzler, K. Deb, and L. Thiele, "Comparison of multiobjective evolutionary algorithms: Empirical results," *Evolutionary computation*, vol. 8, no. 2, pp. 173–195, 2000.

[28] T. Hastie, R. Tibshirani, and J. Friedman, *The elements of statistical learning*. Springer series in statistics Springer, Berlin, 2001, vol. 1.

[29] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning representations by back-propagating errors," *Cognitive modeling*, vol. 5, no. 3, p. 1, 1988.

[30] G. Cybenko, "Approximation by superpositions of a sigmoidal function," *Mathematics of control, signals and systems*, vol. 2, no. 4, pp. 303–314, 1989.

[31] K. Hornik, M. Stinchcombe, and H. White, "Multilayer feedforward networks are universal approximators," *Neural networks*, vol. 2, no. 5, pp. 359–366, 1989.

[32] K. Hornik, "Approximation capabilities of multilayer feedforward networks," *Neural networks*, vol. 4, no. 2, pp. 251–257, 1991.

[33] M. Li and P. Vitányi, *An introduction to Kolmogorov complexity and its applications*. Springer Science & Business Media, 2013.

[34] G. J. Chaitin, A. Arslanov, and C. Calude, "Program-size complexity computes the halting problem," Department of Computer Science, The University of Auckland, New Zealand, Tech. Rep., 1995.

[35] A. M. Turing, "On computable numbers, with an application to the entscheidungsproblem," *Proceedings of the London mathematical society*, vol. 2, no. 1, pp. 230–265, 1937.

[36] W. Teahan, "Probability estimation for ppm," in *In Proceedings NZCSRSC'95. Available from http://www. cs. waikato. ac. nz/wjt.* Citeseer, 1995.

[37] C. Bloom, "Ppmz2: high-compression markov predictive coder, 1999."

[38] D. Ienco, R. G. Pensa, and R. Meo, "From context to distance: Learning dissimilarity for categorical data clustering," *ACM Transactions on Knowledge Discovery from Data (TKDD)*, vol. 6, no. 1, pp. 1–25, 2012.

[39] E. Aldana-Bobadilla and A. F. Kuri-Morales, "A clustering method based on the maximum entropy principle," *Entropy*, vol. 17, no. 1, pp. 151–180, 2015.

[40] E. T. Jaynes, "Information theory and statistical mechanics," *Physical review*, vol. 106, no. 4, pp. 620–630, 1957.

[41] T. Kohonen, "Essentials of the self-organizing map," *Neural networks*, vol. 37, pp. 52–65, 2013.

[42] K. Toutanova and C. D. Manning, "Enriching the knowledge sources used in a maximum entropy part-of-speech tagger," in *Proceedings of the 2000 Joint SIGDAT conference on Empirical methods in natural language processing and very large corpora: held in conjunction with the 38th Annual Meeting of the Association for Computational Linguistics-Volume 13*. Association for Computational Linguistics, 2000, pp. 63–70.

[43] K. Toutanova, D. Klein, C. D. Manning, and Y. Singer, "Feature-rich part-of-speech tagging with a cyclic dependency network," in *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology-Volume 1*. Association for Computational Linguistics, 2003, pp. 173–180.

[44] C. D. Manning, M. Surdeanu, J. Bauer, J. Finkel, S. J. Bethard, and D. McClosky, "The Stanford CoreNLP natural language processing toolkit," in *Association*

for Computational Linguistics (ACL) System Demonstrations, 2014, pp. 55–60. [Online]. Available: http://www.aclweb.org/anthology/P/P14/P14-5010

[45] M. Recasens and M. A. Martí, "Ancora-co: Coreferentially annotated corpora for spanish and catalan," *Language resources and evaluation*, vol. 44, no. 4, pp. 315–345, 2010.

[46] G. Leech and A. Wilson, "Recommendations for the morphosyntactic annotation of corpora eagles report," 1996.

[47] U. Mejía, "CENG," Tesis, Instituto Tecnológico Autónomo de México, Río Hondo No.1, Col. Progreso Tizapán, 01080 Ciudad de México, México, 2016.

[48] J. J. Grefenstette, "Optimization of control parameters for genetic algorithms," *IEEE Transactions on systems, man, and cybernetics*, vol. 16, no. 1, pp. 122–128, 1986.

[49] A. Kuri-Morales and J. Sagastuy-Breña, "A parallel genetic algorithm for pattern recognition in mixed databases," in *Mexican Conference on Pattern Recognition*. Springer, 2017, pp. 13–21.

[50] R. A. Fisher, "The use of multiple measurements in taxonomic problems," *Annals of eugenics*, vol. 7, no. 2, pp. 179–188, 1936.

[51] R. L. Thorndike, "Who belongs in the family?" *Psychometrika*, vol. 18, no. 4, pp. 267–276, 1953.

[52] C. Reilly, C. Wang, and M. Rutherford, "A rapid method for the comparison of cluster analyses," *Statistica Sinica*, pp. 19–33, 2005.