Instituto Tecnológico de Costa Rica
Escuela de Ingeniería en Computación
Programa de Maestría

Evaluación del efecto en el algoritmo de Análisis Semántico Latente al utilizar colecciones de datos cada vez más grandes para la detección y extracción de sinónimos y su independencia respecto al lenguaje, por medio de su implementación distribuida

Tesis para optar por el grado de

Magister Scientiæ en Computación con énfasis en Ciencias de la Computación

Autor:
Rafael Alfaro Flores

Asesor:
José E. Araya Monge, Ph.D.

Cartago, Costa Rica
Junio, 2014

Abstract

Access to large data, especially for text processing applications, results in more effective algorithms and therefore becomes transcendental to take advantage of these large amounts of data. Latent Semantic Analysis (LSA) is an unsupervised machine learning algorithm which benefits from these features and can be used for synonym detection and extraction. LSA takes advantage of the implicit semantic structure that exists in the association between documents and the terms they contain to statistically analyze the relationships between the terms of the collection of text documents; and because it uses a strictly mathematical approach, it is inherently independent of language. This is a thesis for the Masters in Computing degree that analyzes the LSA algorithm in a distributed environment, in order to evaluate its effect for synonym detection and extraction on larger collections of data.

Resumen

El acceso a una mayor cantidad de información, especialmente para aplicaciones de procesamiento de texto, se traduce en algoritmos más eficaces y por lo tanto, se vuelve trascendental aprovechar las cantidades abundantes de datos que se tienen a mano. El Análisis Semántico Latente (LSA) es un algoritmo de aprendizaje de máquina no supervisado que se beneficia de estas características y puede ser utilizado para la detección y extracción de sinónimos. LSA se aprovecha de la estructura semántica implícita que existe en la asociación entre documentos y los términos que éstos contienen para analizar de forma estadística las relaciones que existen entre los términos de la colección de documentos de texto, además al basarse en un enfoque estrictamente matemático, es inherentemente independiente del lenguaje. Esta es una tesis para el grado de Maestría en Computación que analiza el algoritmo LSA en un ambiente distribuido, con el objetivo de evaluar el efecto de éste al detectar y extraer sinónimos en colecciones de datos cada vez más grandes.

www.tec.ac.cr

TEC | Tecnológico de Costa Rica
Programa de Maestría en Computación
Teléfono: 2550-2402

APROBACIÓN DE TESIS FINAL

"Evaluación del efecto en el algoritmo de Análisis Semántico Latente al utilizar colecciones de datos cada vez más grandes para la detección y extracción de sinónimos y su independencia respecto al lenguaje, por medio de su implementación distribuida"

TRIBUNAL EXAMINADOR

José Enrique Araya Monge
Profesor Asesor

PhD. José Helo Guzmán
Profesor Lector

José Alberto Aguilar Romero
Profesional Externo

Dr. Roberto Cortés Morales
Coordinador del Programa de Maestría en Computación

Junio, 2014

Contenido

Índice de figuras

Acrónimos

HDFS Hadoop Distribuided File System

LDA Latent Dirichlet Allocation

LSA Latent Semantic Analysis

ML Machine Learning

PLSA Probabilistic Latent Semantic Analysis

PMI-IR Pointwise Mutual Information - Information Retrieval

SSVD Stochastic Singular Value Decomposition

SVD Singular Value Decomposition

TF-IDF Term Frequency - Inverted Document Frequency

TOEFL Test of English as a Foreign Language

CAPÍTULO 1

Introducción y antecedentes

> Humans are better at seeing the connections than any software is, though humans often need software to help.
>
> Jim Stikeleather

1.1 Introducción

El constante y acelerado incremento en la capacidad de las tecnologías de almacenamiento de datos en los últimos 20 años [28] ha propiciado un incremento paralelo en la cantidad de documentos y datos que se manejan y almacenan en sistemas informáticos del mundo real. Según Gantz [14], se estima que el tamaño del "universo digital" era de aproximadamente 0.18 zetabytes (10^{21} bytes) con un crecimiento esperado para el 2011 a 1.8 zetabytes. "Las colecciones grandes de datos son un hecho del mundo, y por lo tanto, son un tema con el que los sistemas del mundo real deben lidiar" [20].

El acceso a una mayor cantidad de información, especialmente para aplicaciones de procesamiento de texto, se traduce en algoritmos más eficaces y por lo tanto se vuelve trascendental aprovechar las cantidades abundantes de datos que se tienen a mano [16; 20; 28]. Con colecciones grandes de datos, la construcción de modelos estadísticos de lenguaje para tareas de análisis de datos textuales no estructurados se torna más sencilla debido a la gran cantidad de aspectos, raros y comunes, del comportamiento humano que se capturan en estas colecciones. En la mayoría de tareas de procesamiento de texto se ha comprobado que resulta más eficiente trabajar con una colección más grande sin anotaciones, que con una colección pequeña con anotaciones y reglas muy específicas [16]. Esto se debe principalmente a que el lenguaje natural es inherentemente complejo, con cientos de miles de palabras en su vocabulario, así como una cantidad considerable de construcciones gramaticales; además, que cada día se establecen nuevas palabras dentro del vocabulario y se modifican los usos anteriores de las existentes.

Al trabajar con texto no estructurado y con una gran cantidad de datos de entrenamiento, las estadísticas evidentes de las palabras y la coocurrencia de las mismas permite estimar modelos en un tiempo proporcional a la cantidad disponible de datos y con frecuencia es posible paralelizar estas tareas. Por lo tanto, este aprendizaje a partir de colecciones grandes de datos es escalable [16].

La extracción y detección de sinónimos en datos no estructurados es una de las tareas de procesamiento de texto que es posible abordar utilizando un modelo de lenguaje estadístico. Para aplicaciones de acceso y procesamiento de texto, la extracción y detección de sinónimos es una herramienta de gran utilidad ya que permite, por ejemplo, normalizar entidades, eliminar términos duplicados y mejorar filtros de búsqueda basados en texto [23; 9].

Existen diccionarios de sinónimos creados a mano para este fin, como por ejemplo *WordNet*[1], pero no siempre es posible utilizarlos, ya que muchos sinónimos son alta-

[1] http://wordnetweb.princeton.edu/perl/webwn

mente dependientes del contexto [16], además, en estos diccionarios se omiten muchos términos científicos y técnicos. Según [26], en un estudio realizado con artículos de publicaciones científicas y técnicas, se detectó que apenas el 70 % de las palabras claves de dichos artículos se encontraban listados en *WordNet*. Otro problema que presenta este enfoque es la cantidad de trabajo que conlleva agregar una nueva palabra al diccionario así como mantenerlo actualizado.

La interpretación semántica en texto no estructurado tiene que lidiar con imprecisiones y ambigüedades propias de los lenguajes naturales, en especial al trabajar con sinónimos, ya que por ejemplo, un mismo significado puede ser expresado de diferentes formas, y a su vez, una misma expresión puede tener diferentes significados, por lo que es necesario poder identificar y diferenciar estos casos. Al traducir este fenómeno a un modelo de lenguaje estadístico para la detección de sinónimos, se distinguen dos propiedades presentes en las colecciones de documentos de textos: la *colocación* que hace referencia a elementos gramaticalmente vinculados que ocurren en un orden particular y la *asociación* o *coocurrencia* que se refiere al fenómeno más general de que las palabras son susceptibles de ser utilizadas en el mismo contexto [26]. Los modelos estadísticos para la detección de sinónimos se basan en la *coocurrencia* de las palabras.

Los modelos estadísticos de lenguaje más simples sufren del problema de dispersión de los datos, ya que su desempeño es bastante deficiente cuando las palabras son poco frecuentes debido a la escasez de datos [26]. Latent Semantic Analysis (LSA) se presenta como una opción para hacerle frente a este problema. Este algoritmo utiliza la técnica Singular Value Decomposition (SVD) del álgebra lineal para analizar las relaciones estadísticas que existen entre las palabras de una colección de texto. A partir de una matriz de relaciones término/documento, ésta se descompone utilizando SVD para obtener un modelo que expone estructuras semánticas latentes al disminuir la dimensionalidad del modelo original [9].

Una de las principales ventajas de trabajar con LSA para la detección de sinónimos

por medio de un análisis semántico es que este algoritmo es inherentemente paralelizable [4], por tanto es posible trabajar en ambientes distribuidos y con colecciones de datos grandes (que sobrepasan las capacidades normales de memoria y procesamiento para su manipulación en un único equipo de cómputo) para lograr un mejor desempeño. Es por esto que resulta natural el utilizar un modelo de programación distribuida como *MapReduce* para la implementación de LSA [8].

MapReduce está diseñado para el procesamiento y generación de colecciones de datos grandes. Una de sus implementaciones, *Hadoop*, provee una plataforma confiable de almacenamiento distribuido (Hadoop Distribuided File System (HDFS)), así como una plataforma de análisis que implementa *MapReduce* [28].

Este documento es una tesis para el grado de Maestría en Computación con énfasis en Ciencias de la Computación que estudia el desempeño de LSA en la detección de sinónimos con colecciones de datos grandes. El objetivo es medir y evaluar el efecto de aplicar el algoritmo con colecciones de datos cada vez más grandes.

En el resto de la sección 1 se incluye la teoría relacionada con LSA y las herramientas de procesamiento distribuido que se utilizaron para su implementación, se presenta el problema y la hipótesis a demostrar, y se presenta un resumen de las investigaciones más relevantes de detección de sinónimos utilizando LSA. En la sección 2 se describen los objetivos de este trabajo, así como sus contribuciones, alcances y limitaciones. La metodología y los resultados obtenidos se detallan en la sección 3 y las conclusiones y recomendaciones en la sección 4.

1.2 Marco teórico

1.2.1 Análisis Semántico Latente

El algoritmo de **Análisis Semántico Latente o LSA** (por sus siglas en inglés) se aprovecha de la estructura implícita de orden superior (o estructura semántica) que existe en la asociación entre documentos de texto y los términos que éstos contienen, para analizar de forma estadística las relaciones que existen entre los términos de una colección de documentos [9; 26]. Este análisis estadístico permite a LSA identificar y distinguir relaciones de sinonimia y polisemia existentes entre los términos de la colección. Al basarse en un enfoque estrictamente matemático, LSA es inherentemente independiente del lenguaje, lo que permite a LSA obtener el contenido semántico de información escrita en diferentes idiomas sin la necesidad de utilizar estructuras auxiliares como por ejemplo diccionarios [29].

La colección de documentos se representa como una matriz de términos-documentos en la cual las filas corresponden a los términos (t), las columnas a los documentos (d) y las celdas representan el número de veces una palabra ocurre dentro de un documento (o algún otro peso como Term Frequency - Inverted Document Frequency (TF-IDF) que se detalla más adelante) [23]. Esta matriz, es entonces analizada por medio de una Descomposición de Valores Singulares o SVD (por sus siglas en inglés) para derivar el modelo de la estructura semántica latente. SVD representa los términos y los documentos como vectores dentro de un espacio de dimensionalidad reducida y se utiliza el producto escalar o el coseno entre dos puntos en el espacio para determinar la similitud de estos puntos [9; 13].

Una matriz A de tamaño $t \times d$ descompuesta con SVD produce tres matrices de la forma:

$$A = T_0 S_0 D_0'$$

donde T_0 y D_0 tienen columnas ortonormales (ortogonales y de tamaño uno) y son las

matrices izquierda y derecha, respectivamente, de vectores singulares y S_0 es una matriz diagonal compuesta de los valores singulares de A como se muestra en la figura 1.1. Los elementos de la matriz diagonal se construyen de forma que todos sean positivos y estén ordenados decrecientemente. Las columnas de T_0 y D_0 son intercambiables y deben ser reordenadas para corresponder al ordenamiento de la diagonal [9]. En el capítulo 9 de [13] se detalla el método para la obtención de cada una de las matrices resultantes de la descomposición de valores singulares de una matriz.

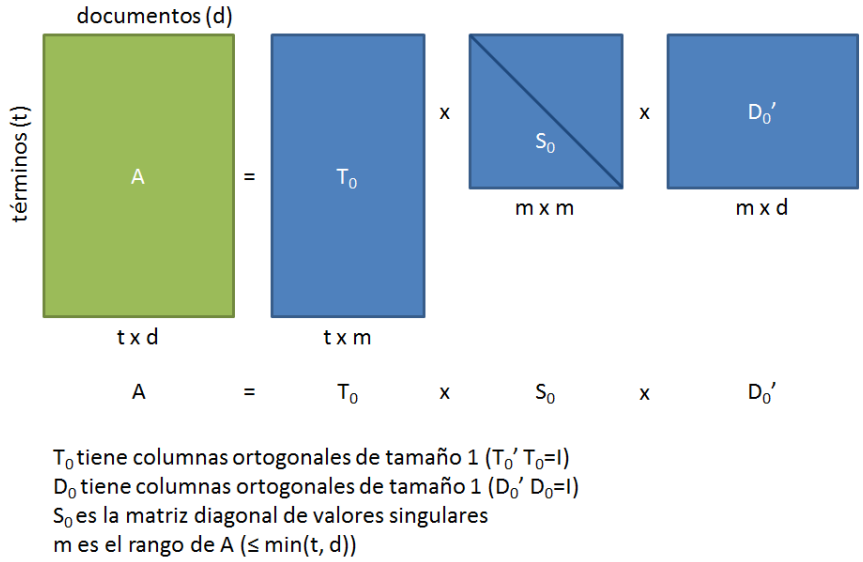


Figura 1.1: SVD
Basado en [9]

Una de las características más importantes de SVD es que permite realizar un ajuste óptimo aproximado usando matrices más pequeñas. Esto se logra trabajando únicamente con los k valores singulares más grandes de S_0; al resto se les asigna 0. El producto de las matrices resultantes es una matriz $\hat{A}$ que se aproxima a A y es de dimensionalidad o rango k. Las matrices entonces son reducidas en tamaño, eliminando las filas y columnas de S_0 que contienen solo ceros, y las columnas correspondientes en T_0 y D_0 como se muestra en la figura 1.2. Se obtiene el modelo reducido:

$$A \approx \hat{A} = TSD'$$

de rango k que aproxima los datos originales [9; 13]. En [15] se describe el método *Block Lanczos* para computar SVD en matrices dispersas grandes (que no pueden ser procesadas en memoria). Además, en [17] se describe un algoritmo estocástico paralelizable para calcular SVD implementado en *Mahout* (biblioteca de algoritmos de aprendizaje de máquina con implementación distribuida).

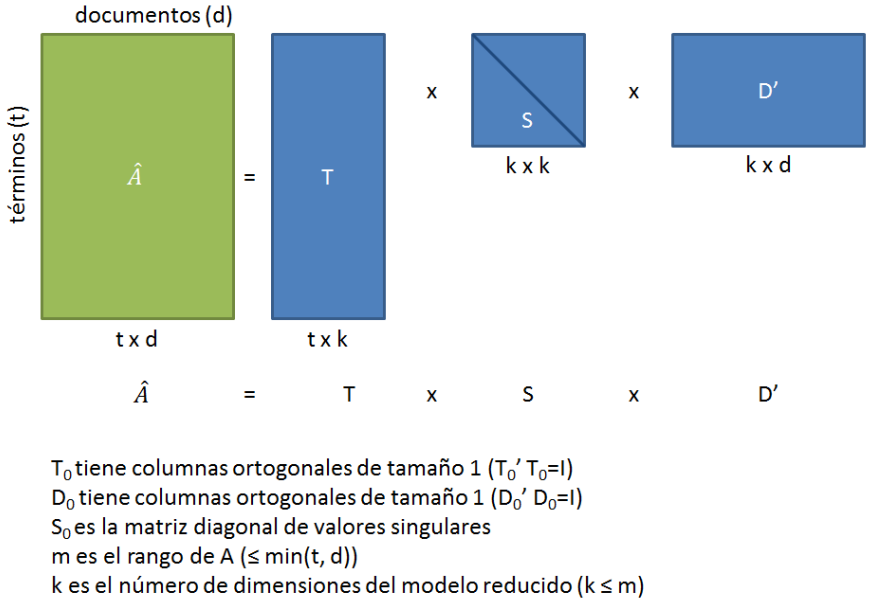


Figura 1.2: SVD Reducido
Basado en [9]

Si el valor de k es muy pequeño, puede ocurrir que no se contemplen todos los datos, pero si es muy grande, se pueden introducir errores en la muestra. Dupert [11] buscó identificar el número óptimo de k para utilizar LSA para la detección de sinónimos. Demostró que trabajando con valores de k en un rango entre 100 y 350 se mantenía la precisión deseada en sus experimentos; al incrementar y sobrepasar el rango de 400 se disminuye la precisión. Deerwester y otros en [9] trabajaron con valores de k en un rango entre 50 y 150 para colecciones significativamente más pequeñas, logrando una mejora en sus resultados al aproximarse al k de 150. Ambos investigadores concuerdan en que se requieren suficientes dimensiones para capturar la estructura real de los términos y documentos, pero no tantas, puesto que el modelo comienza a decaer debido

al ruido introducido por las dimensiones adicionales. El valor final de k va a depender del tipo de aplicación que se esté construyendo y de la calidad de datos que se estén utilizando. Por lo general se trabaja con dimensiones de entre 100 y 300 [9; 26; 11].

Utilizando SVD es posible realizar tres tipos de análisis comparativos entre los datos procesados: comparar lo similares que son dos palabras, comparar lo similares que son dos documentos y comparar cuán asociada está una palabra a un documento [9]. Para la extracción y detección de sinónimos se analizará la similitud entre palabras.

El producto escalar (figura 1.3a) o la similitud del coseno (figura 1.3b) entre dos vectores de fila de $\hat{A}$ determina la similitud del patrón de ocurrencias de dos palabras dentro de la colección de documentos. Como se demuestra en [9], el cálculo de la similitud entre dos palabras utilizando LSA se puede reducir a calcular dicha similitud entre filas de la matriz TS, gracias a las propiedades ortonormales de las matrices T y D resultantes de la descomposición; multiplicar o no por la matriz D' implica únicamente desplazar todos los puntos de la matriz la misma distancia y en la misma dirección.

$$\vec{q} \cdot \vec{d} = \sum_{i=1}^{n} q_i d_i$$

(a) Producto escalar

$$\frac{\vec{q} \cdot \vec{d}}{\|\vec{q}\|\|\vec{d}\|} = \frac{\vec{q}}{\|\vec{q}\|} \cdot \frac{\vec{d}}{\|\vec{d}\|} = \frac{\sum_{i=1}^{n} q_i d_i}{\sqrt{\sum_{i=1}^{n} q_i^2}\sqrt{\sum_{i=1}^{n} d_i^2}}$$

(b) Similitud de coseno

Figura 1.3: Medidas de similitud para LSA

1.2.2 Ambiente para procesamiento distribuido

Para poder aplicar el algoritmo LSA se debe tener como entrada una matriz documentos x términos. Para el cálculo de las entradas de dicha matriz se requiere indexar la colección de documentos: separar documentos en términos y calcular los pesos de los términos en cada documento [9]. El proceso de indexación es realizado por medio de las

plataformas *Aspire* y *Solr/Lucene* que se detallan a continuación. Para la implementación del algoritmo LSA se utilizará una herramienta de la biblioteca de aprendizaje de máquina *Mahout* que corre en la plataforma *Hadoop* que se explican a continuación.

Aspire

Aspire, desarrollado por *Search Technologies Corp.*[2], es una plataforma que provee un marco de trabajo para la adquisición de documentos y registros de datos de prácticamente cualquier fuente de datos, su procesamiento o enriquecimiento y su posterior publicación a aplicaciones que utilizarán la información procesada, como por ejemplo motores de búsqueda. *Aspire* trabaja como una aplicación intermediaria entre la fuente de datos y la aplicación destino. El acceso a los documentos o registros se realiza por medio de *streaming*[3] y no modifica los documentos en la fuente de datos, ni se almacenan localmente para su procesamiento, a menos que se especifique explícitamente que este sea su comportamiento [2].

Solr y Lucene

Lucene es una poderosa biblioteca de recuperación de información de alto desempeño y escalable. Es un proyecto de código abierto escrito en Java, parte de *Apache Software Foundation*. *Lucene* provee la funcionalidad necesaria para crear índices y componentes de búsqueda para estos índices. La información almacenada en un índice de *Lucene* permite acceder de forma sencilla el peso denominado **TF-IDF** de una palabra dentro del índice [22].

TF-IDF es un esquema que busca un balance entre las ocurrencias locales y globales de las palabras dentro de los documentos de la colección y se define de en la fórmula de la figura 1.4, donde $tf_{t,d}$ es la frecuencia del término t en el documento d y df_t es

[2]http://www.searchtechnologies.com/

[3]Distribución de datos a través de una red de computadoras de manera que el cliente consume el producto al mismo tiempo que se descarga.

$$w_{t,d} = (1 + \log(tf_{t,d}) \times \log_{10}(N/df_t)$$

Figura 1.4: Fórmula para calcular *TF-IDF*

el numero de documentos en los que aparece el término t; N es la cantidad total de documentos en la colección.

Solr[4], parte del proyecto *Apache Lucene*^{TM}, es una plataforma de búsqueda que utiliza *Lucene* como su núcleo y que provee interfaces de programación de aplicaciones (API por sus siglas en inglés) que facilitan la integración de otras aplicaciones con *Lucene* como es el caso de *Aspire*. A través de la interfaz HTTP/XML[5] de *Solr* se publicarán desde *Aspire* los documentos que serán indexados.

MapReduce

MapReduce es un modelo de programación funcional para expresar computaciones distribuidas en cantidades masivas de datos utilizando para su procesamiento clústers de servidores/computadoras comunes [8; 20]. *MapReduce* representa el primer paso ampliamente adoptado que se aleja del modelo von Neumann; puede ser visto como el primer gran avance en la búsqueda de nuevas abstracciones que permiten organizar computaciones, no sobre computadoras individuales, sino sobre clústers enteros [20].

El cómputo de *MapReduce* (ver figura 1.5) toma una pareja de entrada de la forma identificador/valor y produce un conjunto de salida de pares de la forma identificador/valor y se basa en dos funciones escritas por el usuario: *map* y *reduce*. *Map* toma una pareja entrada y produce un conjunto de pares identificador/valor como una salida intermedia. *MapReduce* agrupa todos los valores intermedios asociados a un mismo identificador y se los envía a la función de *reduce*. *Reduce* recibe como entrada un identificador y el conjunto de valores asociados a este identificador y ejecuta una función de agregación para obtener generalmente un conjunto más pequeño de valores [8; 20].

[4]http://lucene.apache.org/solr/
[5]http://lucene.apache.org/solr/4_8_1/tutorial.html# Indexing+Data

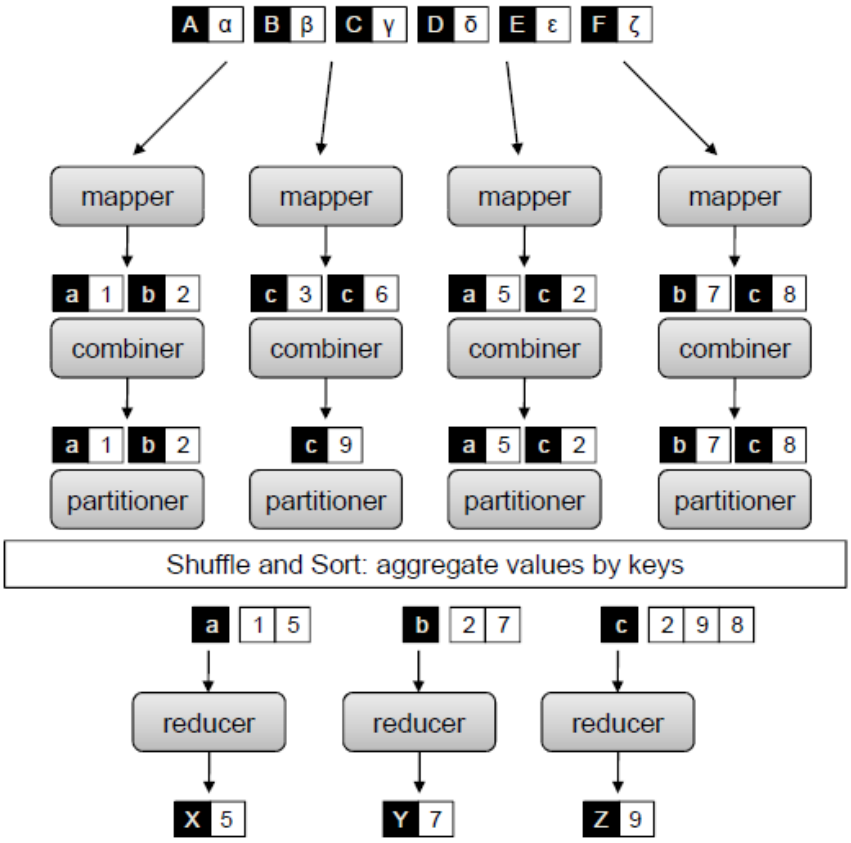


Figura 1.5: Modelo Completo de *MapReduce*
Los *combiners* se encargan de combinar resultados de forma local en los nodos de los *mappers*, los resultados combinados locales son enviados a los *partitioners* que definen a cuales *reducers* deben ser enviados los resultados basados en el valor de la llave (*combiners* y *partitioners* son componentes opcionales). Tomado de [8]

Hadoop y *Mahout*

Hadoop[6] es una plataforma para computación distribuida a gran escala que soporta las complejidades de paralelización, tolerancia a fallos y balanceo de carga. Es un proyecto de código abierto, parte de *Apache Software Foundation*.

Provee un sistema de archivos distribuido: *Hadoop Distributed Filesystem* (HDFS) diseñado para el almacenamiento de archivos muy grandes y con acceso de solo lectura a los datos por medio de *streaming*. Utiliza replicación y redundancia para un acceso más eficiente a los datos (mantiene 3 copias de cada bloque en nodos diferentes) [28]. *Ha-*

[6]http://hadoop.apache.org/

doop dispone de dos servicios llamados *namenode* y *datanode*, los cuales son encargados de administrar el sistema de archivos distribuidos. Los servicios *namenode* mantienen actualizados los metadatos de los archivos, mientras que los servicios *datanode* almacenan los bloques de datos de los archivos. La figura 1.6 tomada de [19] ilustra dichos servicios.

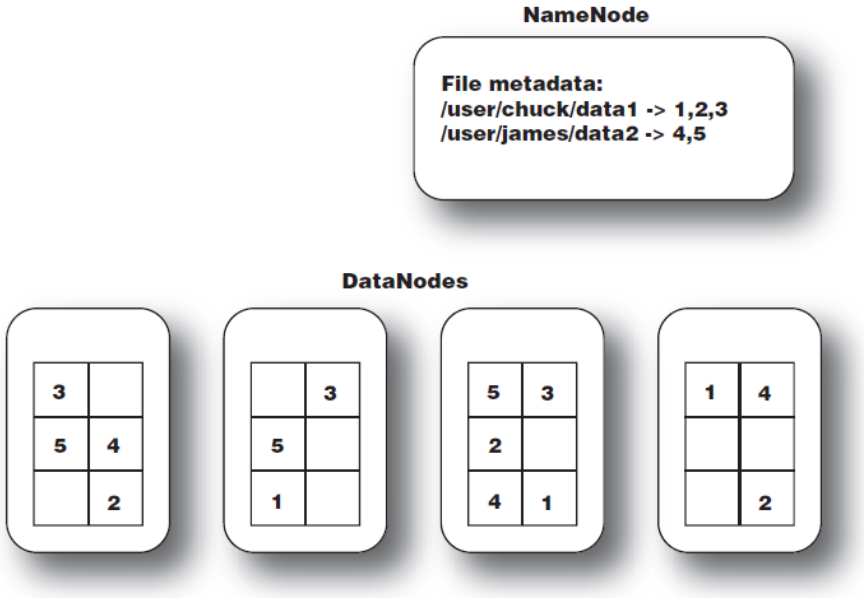


Figura 1.6: Diagrama de Servicios de HDFS
tomado de [19]

Hadoop proporciona además una plataforma para la ejecución de aplicaciones programadas con el modelo de programación *MapReduce* [28]. En la figura 1.7 se muestra un diagrama con los servicios de *Hadoop* encargados de ejecutar aplicaciones *MapReduce*: el*jobtracker* (encargado de particionar las tareas de *map* o *reduce*) y los *tasktrackers* (encargados de ejecutar las diferentes tareas de *map* o de *reduce* o ambas).

Hadoop es la plataforma sobre la cual ejecuta *Mahout*[7], una biblioteca de aprendizaje de máquina (o Machine Learning (ML)), de código abierto y parte también, de *Apache Software Foundation* [1]. *Mahout* provee implementaciones de una serie de algoritmos escalables y distribuidos de ML, entre ellos, dos implementaciones de SVD que se

[7]http://mahout.apache.org/

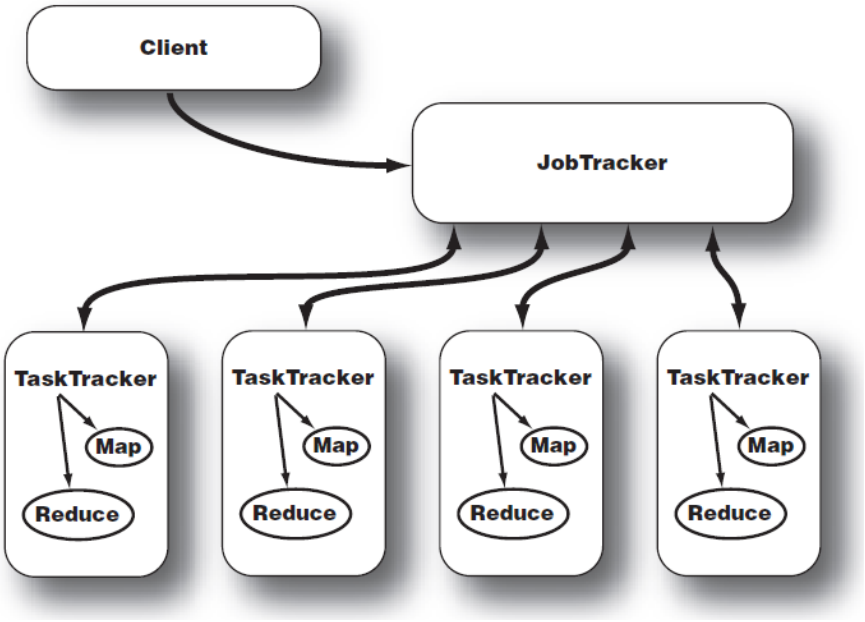


Figura 1.7: Diagrama de Servicios de *MapReduce*
tomado de [19]

analizarán como parte de esta investigación: *Stochastic Singular Value Decomposition* [5] y una versión distribuida del algoritmo de *Block Lanczos* para computar SVD [4].

1.3 Descripción del problema

Se quiere aprovechar la estructura implícita de orden superior, o estructura semántica, que existe en la asociación de términos con respecto a los documentos que los contienen, por medio de LSA, con el fin de identificar sinónimos de un término utilizando como medida la proximidad de los términos (coseno del ángulo entre los términos) dentro del espacio de dimensionalidad reducida modelado.

Además, se desea minimizar el problema de la dispersión de los datos en la matriz de entrada de LSA, incrementando el volumen de los datos de entrenamiento, con el fin de mejorar el desempeño del algoritmo para detectar sinónimos, por medio de la proximidad entre términos [26]. A la vez, buscar escalar el algoritmo de LSA para aceptar grandes volúmenes de datos, sin perder la simplicidad de su implementación, aprovechando la paralelización inherente de SVD [4].

1.4 Hipótesis

Si se entrena LSA con colecciones de datos cada vez más grandes, entonces se logrará incrementar el porcentaje general de sinónimos detectados correctamente, debido a que se construye un mejor contexto de evaluación. “Las técnicas estadísticas típicamente sufren del problema de la dispersión de datos: su desempeño es pobre cuando las palabras son relativamente raras, debido a la escasez de datos” [26].

1.5 Antecedentes

Turney [26] presenta un estudio comparativo entre Pointwise Mutual Information - Information Retrieval (PMI-IR) y LSA como algoritmos para la evaluación de sinónimos. PMI-IR es un algoritmo basado en la coocurrencia de palabras y utiliza como colección de entrenamiento el índice de un motor de búsqueda Web (el autor utilizó Al-

taVista). Los experimentos realizados consistían en tomar una palabra base y seleccionar de una lista de cuatro posibles soluciones la palabra que correspondía a un sinónimo de la palabra base.

PMI-IR evaluaba las soluciones utilizando una fórmula basada en la cantidad de resultados que retorna el motor de búsqueda para la palabra base y la posible solución siendo evaluada: $puntaje(\mathbf{posible}) = resultados(\mathbf{base} NEAR \mathbf{posible})/resultados(\mathbf{posible})$, con algunas variantes de esta fórmula. Mientras que para LSA se calculaba el coseno del ángulo entre los dos términos. PMI-IR trabajó con una colección de alrededor de 350 millones de documentos, mientras que LSA se entrenó con una colección de alrededor de 30 mil documentos. Los resultados obtenidos fueron los siguientes:

- Ambos algoritmos obtuvieron resultados casi idénticos evaluando LSA contra el caso más básico de PMI-IR.
- La variación de PMI-IR con mejor resultado obtuvo un desempeño 10 % mayor a LSA.
- La interpretación de los resultados es difícil debido a la gran diferencia en la cantidad de datos utilizados entre un algoritmo y otro, y plantea la posibilidad de evaluar el desempeño de LSA con una colección similar a la utilizada para PMI-IR.

Kumaran y otros en [18] utilizaron MindNet[8], un sistema automático de extracción de ontologías para texto no estructurado en inglés, para la extracción de relaciones entre palabras del texto (A sinónimo de B) y utilizaron LSA para desambiguación de significados de las palabras. Compararon los resultados obtenidos de estos experimentos con el diccionario WordNet[9], el cual sí está organizado por el significado de las palabras. Entre sus principales resultados destacan:

[8] http://research.microsoft.com/en-us/projects/mindnet/
[9] http://wordnetweb.princeton.edu/perl/webwn

- Utilizaron el ángulo del coseno como medida de similitud (proximidad) entre dos palabras para distinguir sus significados y se utilizaron umbrales de entre 0.8 y 0.95. Entre mayor el umbral, la cantidad de palabras por conjunto de sinónimos disminuyó, mejorando el resultado final, pero en algunos casos separando conjuntos que no debieron ser separados.

- LSA fue determinante para introducir significado a los conjuntos de palabras extraídas por MindNet y eliminar de estos conjuntos palabras que dentro del contexto analizado no eran en realidad sinónimos del resto de palabras en el conjunto.

- Utilizaron MindNet, no para demostrar que se requiere un *parser* de este tipo para la extracción de sinónimos, sino para demostrar que era posible combinar esta técnica con LSA para producir automáticamente una colección de sinónimos con una calidad similar a la de WordNet (creada a mano).

Oates y otros en [23] desarrollaron un algoritmo de dos fases basado en LSA para la detección de diferentes nombres para una misma entidad. En esta investigación, los experimentos se realizaron con una colección de datos bastante pequeña (77 artículos de noticias con un promedio de 516 palabras por artículo). El objetivo del experimento era demostrar que si se tomaba una palabra (entidad), y ésta se reemplazaba por dos valores distintos, el algoritmo, al buscar alias para el primer valor debería devolver el segundo valor en lo más alto de la lista de sus resultados. Realizaron pruebas con LSA que no les resultaron satisfactorias para encontrar alias, aunque si concluyeron, como es esperado, que LSA si estaba encontrando palabras semánticamente relacionadas. Con el algoritmo de dos fases lograron determinar similitud ontológica considerando el contexto local de las ocurrencias de las palabras resultantes de la ejecución inicial de LSA. Inicialmente se consideró esta técnica para la evaluación de los experimentos de detección de sinónimos, sin embargo, debido al tipo de colecciones de evaluación con las

que se trabajó (preguntas de selección única de encontrar el sinónimo de una palabra), esta técnica finalmente no fue utilizada.

Por último, Dupert en [11] aplicó LSA a una matriz de covarianza de términos en lugar de aplicarla a una matriz de términos/documentos, esto debido a que la dimensión de la matriz de covarianza depende únicamente de la cantidad de términos y no del número de documentos. Los experimentos presentados buscaban determinar el número de factores ortogonales indicado para la correcta detección de sinónimos dentro del cuerpo de documentos utilizados. Dichos experimentos demostraron que el concepto que corresponde a un término específico es capturado por los primeros factores ortogonales hasta un umbral definido y que el resto de factores ortogonales ayudan a distinguir entre el término y sus sinónimos. También concluyeron que el número óptimo de factores ortogonales depende de la consulta así como de la colección de documentos.

CAPÍTULO 2

Objetivos y contribuciones

> The most significant idea for big data is that it allows you to see around corners and react.
>
> Michael Cavaretta

2.1 Objetivo general

Evaluar el efecto en el algoritmo de LSA de utilizar colecciones de datos cada vez más grandes para la detección y extracción de sinónimos y su independencia respecto al lenguaje, por medio de su implementación distribuida.

2.2 Objetivos específicos

- Comparar la eficiencia y el volumen de datos soportado en las implementaciones de SVD estocástica y *Block Lanczos* de *Mahout*.
- Determinar la influencia del tamaño de la colección de datos en la precisión de los resultados obtenidos al detectar y extraer sinónimos mediante una implementación distribuida de LSA.
- Analizar la independencia de LSA con respecto al lenguaje utilizando colecciones de datos en idioma español y en idioma inglés.

2.3 Contribuciones

La investigación llevada a cabo para este trabajo produjo una recopilación de las diferentes técnicas utilizadas en trabajos anteriores para la detección de sinónimos usando LSA; dando como resultado un método paralelizable y distribuido para el cálculo de LSA que aprovecha los puntos positivos de investigaciones previas, y que cumple con los objetivos planteados en este documento.

El diseño e implementación del modelo de LSA distribuido, así como el método de evaluación y detección de sinónimos, usaron herramientas de código abierto, por lo que podrán ser utilizados por otros investigadores y desarrolladores en trabajos y aplicaciones futuras.

Esta investigación buscó determinar cuánto más eficaz resulta trabajar con LSA con volúmenes grandes de datos aportando así un marco de referencia para compararlo contra algoritmos de agrupación más complejos que ya han sido probados y evaluados con colecciones grandes de datos.

2.4 Alcances y limitaciones

Este trabajo se basó en el algoritmo original de LSA descrito por Deerwester y otros en [9]. Para la implementación distribuida de LSA se utilizó *Hadoop* como plataforma del modelo de programación de *MapReduce*.

Como base para *LSA* se trabajó con la implementación de *SVD* Distribuido provista por la biblioteca de código abierto *Mahout*, la cual corre sobre *Hadoop*. Se utilizó además el algoritmo de indexado de *Lucene* para la generación de la matriz de documentos x términos que se usa de entrada para SVD.

Las colecciones de datos de entrenamiento y pruebas se basaron en el contenido completo de la enciclopedia en línea Wikipedia en sus versiones en inglés[1] y en español[2]. Para la colección de verificación de inglés se utilizaron preguntas de sinónimos de exámenes Test of English as a Foreign Language (TOEFL). La colección de verificación para español se construyó a partir de la traducción de las preguntas de la colección en inglés.

Esta investigación se enfocó únicamente en el uso del algoritmo LSA para la detección de estructuras latentes en texto no estructurado y no contempló el uso de otros algoritmos como:

- Probabilistic Latent Semantic Analysis (PLSA). Ding en [10] presenta una variante de LSA en la que plantea un modelo probabilístico basado en la similitud de conceptos aportada por LSA, caracterizando cuantitativamente las asociaciones semánticas de los términos.
- Latent Dirichlet Allocation (LDA). Blei y otros en [7] presentan un modelo bayesiano de tres niveles en el cual cada término dentro de una colección de documentos es modelado como una mezcla finita de tópicos implícitos en los documentos, encontrando de esta forma relaciones entre los términos.

[1] http://dumps.wikimedia.org/enwiki/latest/ descargada el 3 de setiembre de 2013
[2] http://dumps.wikimedia.org/eswiki/latest/ descargada el 27 de febrero de 2014

Por último, se analizó el desempeño de LSA para la detección y extracción de sinónimos y no se tomaron en cuenta otros usos que se le han dado a este algoritmo:

- Indexado y recuperación de información en motores de búsqueda.
- Clasificación y agrupado de documentos.

CAPÍTULO 3

Metodología y resultados

> Essentially, all models are wrong,
> but some are useful.
>
> George E. P. Box

El algoritmo de LSA utiliza SVD como base para analizar relaciones estadísticas que existen entre los términos de una colección de documentos. El primer paso para lograr esto es construir una matriz A en la cual las filas representan los términos y las columnas representan los documentos a los que pertenece un término [9; 26]. Dentro de esta matriz, cada celda tiene asociada un valor que corresponde al peso de un término dentro de un documento. Se utilizará TF-IDF como medida para los pesos dentro de la matriz. La matriz será creada como un índice de *Lucene*[3; 22] y será construida a partir de colecciones de artículos extraídos de Wikipedia.

El siguiente paso es aplicar SVD a la matriz de tamaño $t \times d$ para descomponerla en el producto de tres matrices de la forma $T_0 S_0 D_0'$. Utilizando el algoritmo paralelizable SVD de *Mahout* se procesará de forma distribuida sobre un *clúster* de *Hadoop* la des-

composición de la matriz A. La matriz S_0 producto de la descomposición será reducida como se detalla en [11] a los k factores más grandes. Se eliminan las columnas de T_0 y D_0 que no corresponden a los factores seleccionados de S_0. Se obtiene entonces un modelo reducido $A \approx \hat{A} = TSD'$.

Con el modelo reducido se procede a calcular la distancia entre términos para determinar la similitud entre los mismos. Para comparar dos términos, se calcula la similitud del coseno del ángulo entre los términos a evaluar de la matriz TS [9]. Como se justificó en la sección 1.2.1, no es necesario multiplicar por D' ya que esto simplemente desplaza los puntos la misma distancia y la misma dirección, por lo que afecta a todas las similitudes por igual.

3.1 Herramientas utilizadas

3.1.1 Aspire

Aspire, como herramienta para procesamiento de texto no estructurado, se utilizó para la extracción y preparación del contenido de *Wikipedia* como datos de entrada de la colección de entrenamiento del algoritmo LSA.

Aspire fue configurado para leer y extraer de los archivos de respaldo de *Wikipedia* o *dumps* en sus versiones en inglés[1] y en español[2] cada uno de los artículos que conforman *Wikipedia* en cada uno de estos idiomas. Por cada artículo, *Aspire* se encargó de limpiar el contenido del artículo, extrayendo el texto de las diferentes secciones del artículo (encabezados, tablas, enlaces internos, etc) y removiendo de éste, etiquetas de html y mediawiki que no forman parte del contenido del artículo, así como removiendo en la medida de lo posible términos que no fueran considerados como palabras válidas del idioma correspondiente (concatenaciones de caracteres con secuencias 3 o más repeticiones de una misma letra, concatenaciones de caracteres donde no hubiera presente

[1] http://dumps.wikimedia.org/enwiki/latest/
[2] http://dumps.wikimedia.org/eswiki/latest/

ninguna vocal; incluyendo 'y' como vocal para inglés, etc).

Una vez extraído el contenido, *Aspire* generó una entrada por cada artículo, la cual que fue enviada al motor de búsqueda para ser indexada como un documento independiente.

La figura 3.1 detalla la configuración de *Aspire* utilizada para el procesamiento de artículos de *Wikipedia*.

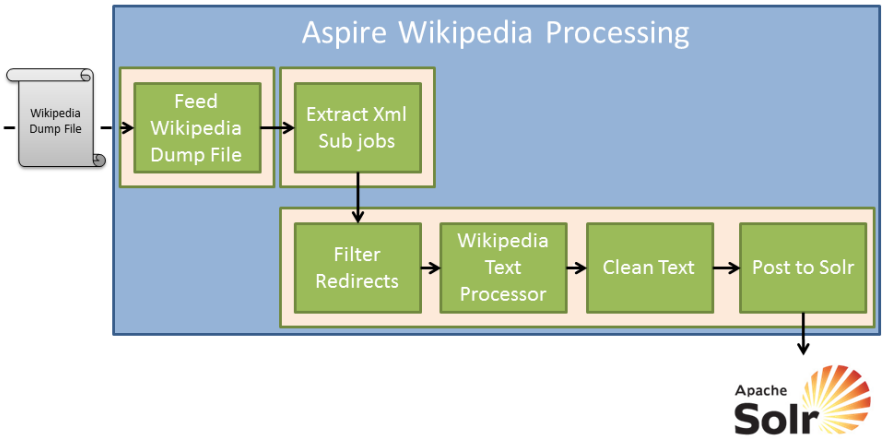


Figura 3.1: Diagrama de Procesamiento de Aspire

3.1.2 Solr y Lucene

Se utilizó *Solr* para la creación de la matriz de términos de entrada debido a la facilidad de integración con *Aspire*[3], y de su núcleo, *Lucene*, con el algoritmo *SSVD* de *Mahout* [5]. La tarea encomendada a *Solr/Lucene* fue la de la construcción de una matriz de términos contra documentos en cuyas celdas se calcularon los valores de TF-IDF para cada intersección de término y documento. Para lograr esto, se configuró un campo de texto en el índice con la propiedad *termVector* habilitada, la cual le indica al motor

[3]https://wiki.searchtechnologies.com/index.php/Post_HTTP

de búsqueda que genere vectores para cada término con el TF-IDF correspondiente al término contra todos los documentos presentes en el índice.

Para la creación del índice y de los vectores de términos se utilizaron las configuraciones que se muestran en las figuras 3.2 y 3.3. El filtro *LowerCaseFilterFactory* transforma todos los términos a minúscula: no se diferencia entre mayúsculas y minúsculas. Con el filtro *StopFilterFactory* se remueven las palabras más cómunes o *stopwords* del índice. Las definiciones de los tipos de los campos fueron modificadas para remover los filtros de *stemming*[4]. Como se detallará más adelante, hacer uso de *stemming* introdujo ruido en los datos. Además, por medio del parámetro *termVector* en la definición del campo *text* se le indica a *Lucene* que genere y almacene los pesos de TF-IDF para cada término indexado y que serán utilizado posteriormente por *Mahout*.

```
<field name="text" type="text_en_splitting" indexed="true" stored="true" multiValued="
    true" termVectors="true"/>
```

(a) Configuración de campo de texto

```
<fieldType name="text_en_splitting" class="solr.TextField" positionIncrementGap="100"
    autoGeneratePhraseQueries="true">
  <analyzer type="index">
    <tokenizer class="solr.WhitespaceTokenizerFactory"/>
    <filter class="solr.StopFilterFactory" ignoreCase="true" words="lang/stopwords_en.
        txt" enablePositionIncrements="true"/>
    <filter class="solr.WordDelimiterFilterFactory" generateWordParts="1"
        generateNumberParts="1" catenateWords="1" catenateNumbers="1" catenateAll="0"
        splitOnCaseChange="1"/>
    <filter class="solr.LowerCaseFilterFactory"/>
    <filter class="solr.KeywordMarkerFilterFactory" protected="protwords.txt"/>
  </analyzer>
</fieldType>
```

(b) Configuración del tipo de campo

Figura 3.2: Configuraciones de *Solr* para colección en inglés

3.1.3 Hadoop

Hadoop proporcionó la plataforma de almacenamiento distribuido y procesamiento paralelizable distribuido sobre la cual se ejecutó y analizó el algoritmo de LSA [28]. Se

[4]Es un método para reducir una palabra a su raíz o lema

```
<field name="text" type="text_es" indexed="true" stored="true" multiValued="true"
    termVectors="true"/>
```

(a) Configuración de campo de texto

```
<fieldType name="text_es" class="solr.TextField" positionIncrementGap="100">
  <analyzer>
    <tokenizer class="solr.StandardTokenizerFactory"/>
    <filter class="solr.LowerCaseFilterFactory"/>
    <filter class="solr.StopFilterFactory" ignoreCase="true" words="lang/stopwords_es.
        txt" format="snowball" enablePositionIncrements="true"/>
  </analyzer>
</fieldType>
```

(b) Configuración del tipo de campo

Figura 3.3: Configuraciones de *Solr* para colección en español

instaló un clúster de *Hadoop* de 8 nodos en un ambiente virtualizado como se detalla en la figura 3.4.

Máquina	CPU	RAM	DD	Servicios
Master	4 proc	8 GB	400 GB	Name Node Job Tracker
Slave1 a Slave7	4 proc	8 GB	300 GB	Data Node Task Tracker

Figura 3.4: Especificaciones del clúster

El Sistema de Archivos Distribuido de *Hadoop* (HDFS por sus siglas en inglés) se utilizó para almacenar las diferentes matrices de términos de entrada, así como las listas de similitud de términos de salida del algoritmo de LSA.

La plataforma de procesamiento distribuido *MapReduce* se utilizó como base para la programación de comandos utilitarios que fueron creados para la manipulación y transformación de los datos en *Hadoop*, y principalmente, para la implementación del algoritmo LSA en conjunto con el algoritmo Stochastic Singular Value Decomposition (SSVD) proporcionado por *Mahout*.

Para el desarrollo de esta investigación, se trabajó con la versión 1.2.1[5] de *Hadoop*.

[5]hadoop.apache.org/docs/r1.2.1/index.html

3.1.4 Mahout

La biblioteca de algoritmos de aprendizaje de máquina *Mahout* [1] se instaló en *Hadoop* con la finalidad de utilizar el algoritmo SSVD[6] de reducción de dimensiones, base del algoritmo de LSA implementado y como se menciona en [17] buscando aprovechar al máximo las características de paralelización que presenta SSVD.

Mahout facilitó además, por medio del utilitario **lucene.vector** de su biblioteca, la extracción de la matriz de término directamente del índice de *Lucene* (núcleo de *Solr*), y su almacenamiento en HDFS, en el formato requerido por el comando que implementa el algoritmo SSVD.

Para el desarrollo de esta investigación, se trabajó con la version 0.9-SNAPSHOT[7] de *Mahout*.

3.2 Diseño de los experimentos y resultados

3.2.1 Colecciones de entrenamiento

Las colecciones de entrenamiento fueron construidas a partir del texto de las colecciones completas de los artículos de *Wikipedia*, tanto para español como para inglés. Se escogió *Wikipedia* como colección de entrenamiento debido a que ya ha mostrado su utilidad en otras investigaciones de procesamiento de lenguaje natural y específicamente con LSA [27; 17]; así como a las características que presenta: de acceso libre, multilenguaje y contenido colaborativo (contenido con vocabulario enriquecido por autores de diferentes culturas y partes del mundo) [6].

El proceso seguido para la creación de las diferentes colecciones de entrenamiento fue el mismo tanto para español como para inglés y es el que se detalla a continuación:

[6] https://mahout.apache.org/users/dim-reduction/ssvd.html

[7] http://archive.apache.org/dist/mahout/0.9/ La versión 0.9-SNAPSHOT ya ha sido liberada en su versión estable

1. De los *dumps*[8][9] de *Wikipedia* (respaldos periódicos de los artículos y demás archivos de Wikipedia disponibles para descarga), se descargaron los archivos *(es—en)wiki-latest-pages-articles(N).xml.bz2* para el procesamiento local de los mismos.

2. Con base en los tamaños totales de cada colección (número de artículos), se definieron subcolecciones para evaluar el impacto de utilizar más datos con el algoritmo. Para la definición de los tamaños de las subcolecciones, se utilizó como factor de decrecimiento $\sqrt[3]{10}$, como se muestra en la figura 3.5, donde **Term6** es la colección más pequeña y **Term0** la colección más grande formada por la colección completa de artículos de *Wikipedia* para cada idioma. Al ser la colección en español significativamente más pequeña que la colección en inglés, se trabajó con colecciones que por su tamaño corresponden al rango entre **Term2** y **Term5**. Las subcolecciones son acumulativas, es decir, TermN contiene los artículos de TermN+1.

Identificador	Factor de decrecimiento	Documentos		Términos	
		Español	Inglés	Español	Inglés
Term6	$\sqrt[3]{10^6} = 100$	– – ––	62683	– – ––	71519
Term5	$\sqrt[3]{10^5} = 46,42$	133691	135046	215406	132660
Term4	$\sqrt[3]{10^4} = 21,54$	288030	290947	291047	183609
Term3	$\sqrt[3]{10^3} = 10$	620542	626827	424991	255272
Term2	$\sqrt[3]{10^2} = 4,64$	1336918	1350458	627307	361212
Term1	$\sqrt[3]{10^1} = 2,15$	– – ––	2909474	– – ––	537041
Term0	$\sqrt[3]{10^0} = 1$	– – ––	6268271	– – ––	835663

Figura 3.5: Tamaños de las subcolecciones

3. Por medio de *Aspire* se extrajo, limpió e indexó el contenido correspondiente a cada subcolección en el índice de *Solr*. Parte del proceso de limpieza de la colección incluyó configurar el campo del índice de *Solr* para excluir *stopwords* (palabras

[8]http://dumps.wikimedia.org/eswiki/latest/
[9]http://dumps.wikimedia.org/enwiki/latest/

irrelevantes) utilizando las listas predeterminadas de *Solr*. Un subconjunto de *stopwords* puede verse en la figura 3.6. Los filtros de *stemming* no se utilizaron debido a que se observó en pruebas iniciales que estos introdujeron ruido en los datos al agrupar palabras por su raíz, perdiendo el contexto de las mismas; no todas las palabras con la misma raíz están relacionadas. Como se explicará más adelante, el *stemming* también causó problemas en las colecciones de evaluación.

Español	Inglés
de	a
la	an
que	and
el	are
en	as
y	at
a	be
los	but
del	by
se	for
las	if
por	in
un	into
para	is
con	it
no	no
...	...

Figura 3.6: Ejemplos de *stopwords*

4. A partir del índice de la subcolección se creó la matriz de TF-IDF de términos por documentos usando el utilitario de *Mahout* **lucene.vector**. Además de la matriz, este utilitario proporciona un diccionario de términos con sus frecuencias de aparición dentro de la colección indexada; este diccionario se utilizó para remover de la matriz todos los términos cuya frecuencia fuera menor a un umbral preestablecido para disminuir la dispersión de los datos (removiendo términos que resultan irrelevantes al aparecer muy pocas veces) y reducir significativamente el tamaño de la matriz haciendo factible la descomposición en un tiempo razonable.

En la figura 3.7 se muestra el conteo de términos obtenidos al remover términos de baja frecuencia utilizando diferentes umbrales. Para escoger los umbrales se corroboró que en la colección más grande para cada idioma todos los términos a evaluar estuvieran presentes en la matriz; los umbrales escogidos fueron de 10 para la colección en inglés y de 3 para la colección en español. En el caso de la colección en español, al estar trabajando con una colección más pequeña que la de inglés, fue necesario remover únicamente términos de frecuencias muy bajas (1-3) para poder mantener dentro de la colección términos que debido a la naturaleza de la colección aparecían en muy pocos artículos.

Idioma	Frecuencia de aparición	
	< 5	< 10
Inglés	1343328	**835663**

Idioma	Frecuencia de aparición	
	< 3	< 10
Español	**627307**	310100

Figura 3.7: Pruebas preliminares: conteo de términos removiendo términos de baja frecuencia

5. Una vez disponible la matriz de términos se procedió a la descomposición de la misma utilizando el algoritmo SSVD con los parámetros que se muestran en la figura 3.8. Siguiendo las recomendaciones de [9; 23; 26] descritas en la sección 1.2.1 se establece para trabajar con sinónimos un rango de dimensión k entre 100 y 300; se requieren suficientes dimensiones para capturar la estructura real de los términos y documentos, pero no tantas, puesto que el modelo comienza a decaer debido al ruido introducido por las dimensiones adicionales. Basado en estas recomendaciones y a varias pruebas preliminares realizadas (ver figura 3.9), se decidió trabajar con un espacio dimensional de 200 (los 200 valores singulares más grandes de la matriz diagonal de la descomposición). De acuerdo a las

comparativas de los algoritmos SVD y SSVD realizadas por [17], la mejor aproximación de SSVD al resultado matemático de SVD (Block Lanzcos) se logra con **powerIter** = 1. Se le indica al algoritmo que se desea la salida tanto de la matriz U (matriz ortogonal de documentos) como de la matriz V (matriz ortogonal de términos). Se obtiene entonces, para la matriz original, las 3 matrices U, S y V de la descomposición de valores singulares.

```
mahout ssvd --input matrizEntrada --output folderDeSalida --rank 200 --reduceTasks 1 --
    tempDir folderTemp --vHalfSigma false --uHalfSigma false --computeV true --computeU
     true --powerIter 1
```

Figura 3.8: Comando SSVD

	$k = 100$	$k = 200$	$k = 300$
$powerIter = 0$	35.8 %	36.9 %	35.8 %
$powerIter = 1$	35.5 %	42.1 %	X

Debido al tiempo de ejecución tan prolongado para la descomposición con $k = 300$ (aproximadamente el doble de $k = 200$, 72 horas) y debido a que no se observó una diferencia poco significativa en sus resultados con respecto a $k = 100$ y $k = 200$, se decidió realizar la prueba con $powerIter = 1$ únicamente para los k de 100 y 200.

Figura 3.9: Pruebas preliminares: selección de dimensión k basadas en Term0 en inglés y utilizando los exámenes 1 y 2

6. El último paso para obtener la matriz de entrada para el algoritmo LSA que se utilizó en los diferentes experimentos, se debe multiplicar la matriz S por la matriz transpuesta V para, como se indica en [9], poder luego comparar la similitud entre dos términos.

En el apéndice 5 se lista el paso a paso de los comandos requeridos para obtener la matriz de entrada.

3.2.2 Colecciones de evaluación

La evaluación de la capacidad del algoritmo LSA para la detección de sinónimos se realizó utilizando preguntas de selección única, las cuales constan de una palabra a

evaluar y cuatro palabras como posibles sinónimos de la palabra a evaluar. Se midió la capacidad del algoritmo para seleccionar la palabra correcta de entre las respuestas.

Cada colección de evaluación se construyó utilizando preguntas de exámenes de práctica para el TOEFL para evaluar las colecciones en inglés y una traducción de estas preguntas para evaluar las colecciones en español. Específicamente se utilizaron tres colecciones de evaluación distintas (colecciones completas en apéndice 5):

- Examen de 50 preguntas tomadas de [25].
- Examen de 112 preguntas tomadas de [21].
- Examen de 223 preguntas tomadas de [12].

Cada colección de evaluación está conformada por un único archivo de texto, donde cada línea es una pregunta con sus respectivas respuestas separadas por un *tab*. La primera palabra en la línea corresponde a la pregunta y la siguiente palabra corresponde a la respuesta correcta, seguida por las demás opciones como se puede ver en la figura 3.10.

Pregunta	Opción Correcta	Opción 2	Opción 3	Opción 4
associates	colleagues	neighbors	superiors	students
brilliant	intelligent	known	professional	popular
census	survey	group	number	newspaper
constructed	built	guaranteed	located	insured
demonstrate	show	guess	unnecessary	describe
...	...	...	...	...

Figura 3.10: Ejemplo de examen de evaluación

Además del problema observado en las colecciones de entrenamiento con *stemming*, se observó en pruebas iniciales que también afectaba a las colecciones de evaluación, ya que provocó que algunas evaluaciones fueran incorrectas en caso de que se estuvieran evaluando palabras con la mismas raíz, como se muestra en la figura 3.11, pues el valor de similitud de ambas sería el mismo.

Tipo	Pregunta	Opción Correcta	Opción 2	Opción 3	Opción 4
Original	**generic**	**general**	cheap	fresh	elderly
Con *Stemming*	**gener**	**gener**	cheap	fresh	elderli

Figura 3.11: Problema de preguntas de examen de evaluación con *stemming*

3.2.3 Comparación de algoritmos *SVD*

Para la comparación y selección del algoritmo *SVD* que se utilizó para el desarrollo de esta investigación se tomaron en cuenta tres factores: tiempo de ejecución, precisión de los resultados y escalabilidad del algoritmo.

Inicialmente estaba previsto realizar el análisis comparativo considerando pruebas de ejecución de los algoritmos *SVD* y *SSVD* de *Mahout*, sin embargo, dado que existían estudios previos desarrollados por Nathan en [17], así como la información presentada en [4; 5] del sitio oficial de *Mahout*, se decidió no repetir esas pruebas y utilizar éstos como referencia para la selección del algoritmo a utilizar. A continuación se detallan los aportes de estos estudios para cada uno de los factores de evaluación.

- **Tiempo de ejecución**: En un ambiente diseñado para la computación iterativa como *Hadoop*, cada vez que un algoritmo accede a los datos, el algoritmo debería extraer la mayor cantidad de información posible limitando la interacción con los datos y minimizando los tiempos de inicialización y los costos de movilización de los datos. Con el algoritmo SVD de *Block Lanczos*, esto no sucede así. Como se describe en [4; 17] el algoritmo procesa únicamente un vector a la vez por cada iteración de la matriz. Esto implica, no solo múltiples pasadas y movimientos de los datos, sino que tampoco hace un uso óptimo de las capacidades de procesamiento disponibles en el clúster. Por su parte SSVD, como se menciona en [5; 17] realiza únicamente dos iteraciones por la matriz (dos iteraciones adicionales por cada iteración extra configurada en el parámetro **powerIter**; **powerIter=1** para los experimentos) y aprovecha las capacidades de procesamiento en paralelo que

brinda *Hadoop*, mejorando el rendimiento del algoritmo y disminuyendo los tiempos de ejecución. Por lo tanto, para los experimentos diseñados y el ambiente con el que se cuenta, SSVD es más rápido que SVD Block Lanczos.

- **Precisión de los resultados**: El algoritmo SVD Block Lanczos provee una excelente precisión en sus resultados a cambio de las múltiples iteraciones que deben realizarse sobre los datos para garantizar esta precisión. La precisión de SSVD con **powerIter=0** no es tan buena, sin embargo, al utilizar **powerIter=1** la precisión es comparable con SVD Block Lanczos a un costo mucho más bajo [17].

- **Escalabilidad del algoritmo**: Para el tipo de experimentos diseñados en esta investigación, la escalabilidad fue el factor de selección más importante. De acuerdo a las pruebas realizadas en [17], el algoritmo SVD Block Lanczos no escala con colecciones grandes de datos debido a su naturaleza iterativa, lo cual implica que la mayoría de los cálculos deben hacerse de manera serial. Además, en estas pruebas se describen problemas de memoria con colecciones relativamente pequeñas, lo cual se preveía como un posible problema en el ambiente de *Hadoop* para estos experimentos debido a las limitaciones de hardware con que se trabajó. El uso de memoria en SSVD es independiente de las dimensiones de la matriz por lo que el algoritmo puede escalar perfectamente con colecciones de datos bastante grandes, lo cual fue demostrado en las pruebas de [17] al procesar una descomposición de valores singulares de rango 100 sobre una matriz cuadrática cuyas dimensiones superaban los 37000000.

Basado en las comparaciones de los factores descritos anteriormente, la escalabilidad y el tiempo de ejecución fueron determinantes para la selección de **SSVD** como el algoritmo a utilizar, amparado en que la precisión de los resultados de ambos algoritmos es comparable.

3.2.4 Evaluación de sinónimos

Para demostrar la capacidad del algoritmo LSA para la detección de sinónimos se construyó una aplicación de *MapReduce* que evalúa y responde cada pregunta de las colecciones de evaluación.

La aplicación para la evaluación de sinónimos recibe como entrada una matriz de términos (una colección **TermN**) y un examen de evaluación y ejecuta el siguiente procedimiento:

1. Dada la palabra a evaluar de una pregunta, se calcula la similitud de ésta contra todas las palabras de la colección *TermN* de entrada y se ordenan las palabras, por su similitud, de mayor a menor para obtener una lista con las palabras más similares a la palabra pregunta en la parte superior de la lista. Dado que cada fila de una matriz de entrada **TermN** está identificada por una palabra y los valores de la fila corresponden a los k pesos más significativos asociados con cada palabra (debido a la reducción de la dimensión de la matriz por SVD), el cálculo de la similitud entre palabras se realiza tomando los vectores de las dos palabras a comparar y se le calcula la similitud de coseno entre estos dos vectores como se muestra en la figura 3.12.

$$\frac{\vec{q}\cdot\vec{d}}{\|\vec{q}\|\|\vec{d}\|} = \frac{\vec{q}}{\|\vec{q}\|}\cdot\frac{\vec{d}}{\|\vec{d}\|} = \frac{\sum_{i=1}^{n} q_i d_i}{\sqrt{\sum_{i=1}^{n} q_i^2}\sqrt{\sum_{i=1}^{n} d_i^2}}$$

Figura 3.12: Cálculo Similitud de Coseno

.

2. Se localizan cada una de las posibles palabras respuesta en la lista de similitudes de la palabra pregunta a evaluar y se registra tanto el valor de similitud de la posible respuesta con respecto a la pregunta, como su posición dentro de la lista.

La figura 3.13 muestra un ejemplo de la lista de similitudes para la palabra en inglés *miserable*.

3. De las cuatro posibles opciones se toma como respuesta correcta la opción que aparece de primera en la lista. En el caso del ejemplo de la figura 3.13, se escoge como respuesta correcta ***unhappy***. Si una de las opciones no aparece en la colección de entrada, la pregunta es marcada como incorrecta puesto que no es posible asegurar que la opción seleccionada es realmente la correcta.

4. Se repiten los pasos 1 al 3 para todas las preguntas del examen a evaluar.

5. El resultado final de la colección de evaluación se calcula basado en un criterio similar al utilizado por [26]:

   - Respuesta correcta = 1 punto.
   - Respuesta incorrecta = $\frac{-1}{3}$ punto.

Basado en los resultados obtenidos al evaluar los exámenes de sinónimos de las colecciones de evaluación con cada una de las subcolecciones de datos, tanto en inglés como en español, se analizó el desempeño del algoritmo al incrementar el tamaño de la colección de entrada.

Para medir el rendimiento del algoritmo LSA al utilizar colecciones de datos cada vez más grandes, se utilizó como punto de comparación los resultados obtenidos al utilizar un algoritmo aleatorio uniforme. Basado en la probabilidad del algoritmo aleatorio uniforme de seleccionar la respuesta correcta y de acuerdo al criterio de puntaje utilizado para evaluar cada examen, este algoritmo obtendría una puntuación de 0, como se muestra en el modelo de la figura 3.14.

Al evaluar los exámenes contra cada una de las matrices de entrada, se fueron obteniendo mejores resultados conforme aumentaba el tamaño de la matriz, tal como se muestra en los gráficos y los datos de la figura 3.15.

Pregunta	Opción 1 (Correcta)	Opción 2	Opción 3	Opción 4
miserable	unhappy	cruel	wrong	miss

(a) Pregunta de ejemplo

Posición	Similitud	Palabra
1	**1**	**miserable**
2	0.939414782	dearly
3	0.93832363	hated
4	0.936810091	regretting
5	0.936681771	disgust
6	**0.936615517**	**unhappy**
7	0.936135772	bewildered
8	0.932901055	amazement
9	0.930964517	grief
...	...	...
317	**0.881256675**	**cruel**
318	0.881199645	inexplicably
319	0.88099479	intently
320	0.880968044	blissfully
321	0.880952116	scorn
...	...	...
22287	**0.570206362**	**wrong**
22288	0.570204407	erek
22289	0.570191572	consciences
22290	0.570191253	couplet
...	...	...
120125	**0.319463915**	**miss**
120126	0.319462979	loitered
120127	0.319461147	bahrampour
120128	0.319460715	monsieur
120129	0.31945976	mastroianni
120130	0.319459129	explanitory
...	..	...

(b) Lista de similitudes con **miserable**

Figura 3.13: Ejemplo lista de similitud

$$\text{Probabilidad Respuesta correcta} = 25\,\%$$
$$\text{Probabilidad Respuesta incorrecta} = 75\,\%$$

$$25\,\% * 1 + 75\,\% * \frac{-1}{3} = 0$$

Figura 3.14: Modelo de Algoritmo Aleatorio Uniforme

Partiendo del hecho de que el puntaje obtenido con el algoritmo aleatorio uniforme es de 0, se observa que aún con la colección más pequeña de datos (*Term6* en inglés y *Term5* en español), el desempeño del algoritmo LSA es superior, con lo cual se demuestra la utilidad de LSA para la detección de sinónimos en estos escenarios.

Otra forma de visualizar el desempeño del algoritmo LSA es por medio del porcentaje de acierto respecto al total de preguntas evaluadas en cada una de las subcolecciones de entrada. Estos porcentajes se muestran en la figura 3.16.

Si bien es cierto, se observa una mejora en el porcentaje de acierto al utilizar colecciones cada vez más grandes y el resultado es mejor que el del algoritmo aleatorio, el resultado obtenido podría mejorar al incrementar la calidad de los datos en las colecciones de entrada, por ejemplo, utilizando colecciones temáticas o usando diccionarios para filtrar términos no existentes del idioma dentro de las colecciones.

Análisis Estadístico de las colecciones de entrenamiento

A pesar de que se observa una mejora en los resultados conforme se incrementan los tamaños de las colecciones de entrada, es importante evaluar si estos resultados con significativos. Para evaluar el número de éxitos (la cantidad de respuestas correctas en las pruebas de sinónimos), Smucker y otros en [24] consideran el test de signo como un buen candidato para determinar la significancia estadística.

El test de Signo consiste en calcular la distribución binomial para la cantidad de cambios positivos obtenidos al evaluar la colección de evaluación en dos colecciones de entrenamiento diferentes.

Para hacer esta evaluación, se parte de la hipótesis:

- H0: Hay menos o iguales cambios positivos que negativos ($p = 0{,}5$)
- H1: Hay más cambios positivos que negativos ($p > 0{,}5$)

Por ejemplo, al evaluar *Term6* con *Term5*, dada la respuesta de una pregunta, si para *Term6* fue incorrecta y para *Term5* correcta, esto se considera un cambio positivo, si la pregunta es correcta en ambas colecciones, este resultado no se toma en cuenta para la prueba y si la respuesta era correcta en *Term6* y luego incorrecta en *Term5*, se considera como un cambio negativo.

Basado en la anterior recomendación, se evaluó la significancia estadística de las colecciones de entrenamiento utilizando el test de Signo. Para los experimentos se utilizó un nivel de significancia de 95 % ($\alpha = 0{,}05$).

Para cada idioma se realizaron dos evaluaciones diferentes: una primera evaluación con los resultados de todas las preguntas de la colección de evaluación y una segunda evaluación únicamente con las preguntas que todas las colecciones de entrenamiento pudieron contestar (si en *Term6* no estaba la palabra de una de las preguntas, el resultado de esta pregunta no se contabilizó).

Tal como se muestra en las figuras 3.17, 3.18, 3.19 y 3.20 se obtiene que:

- En la evaluación de las colecciones de inglés con todas las preguntas se nota el efecto de mejora conforme aumenta el tamaño de la colección, pero solo es significativo a partir de un tamaño mucho mayor (el cambio de *Term3* a *Term0*), a partir de cierto tamaño el efecto parece atenuarse (no fue significativo el cambio de *Term2* a *Term0*).
- Al eliminar las preguntas que *Term6* no pudo evaluar, de las colecciones de inglés, se pierden la mayoría de las diferencias significativas, lo que sugiere que la gran ventaja de incrementar el tamaño de la colección está en la capacidad de manejar

términos que ocurren poco. De modo que puede resolver adecuadamente el tipo de ejercicios usados en los experimentos.

- En ambas evaluaciones de las colecciones de español se observa que con *Term5* contra los demás sí es significativa la mejora debido al tamaño; con *Term4* y *Term3* no es significativa la mejora de *Term2*.

- El comportamiento poco dominante de *Term2* en ambas evaluaciones de las colección de español es parecido al comportamiento de la evaluación de las colecciones de inglés en las colecciones similares en tamaño como se observa al comparar la figura 3.19 con la sección correspondiente en la figura 3.17 y la figura 3.20 con la sección correspondiente en la figura 3.18, lo cual sugiere que para alcanzar un comportamiento dominante como el de *Term0* en inglés se debe incrementar el tamaño de la colección en español.

3.2.5 Independencia del lenguaje

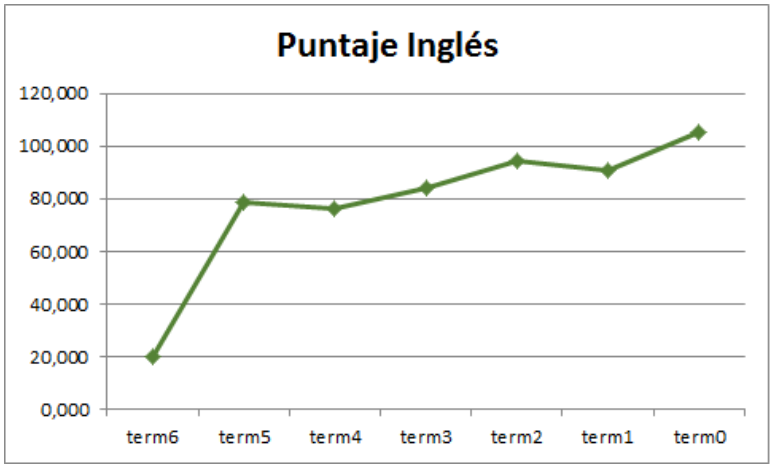
Al realizar los experimentos utilizando colecciones separadas, en inglés y en español, se buscaba comprobar la utilidad del algoritmo LSA para la detección de sinónimos independientemente del lenguaje en el que se estuviera trabajando. Como se muestra en la figura 3.16 de los experimentos realizados, se nota que el desempeño del algoritmo es similar y con resultados favorable en ambos casos, lo cual evidencia que el algoritmo lo que toma en cuenta son las relaciones que existen entre dos puntos (términos) sin importar la semántica de éstos.

Además, se analizó el porcentaje de similitud entre la palabra pregunta y la respuesta seleccionada (no necesariamente la correcta), en todas las preguntas de los exámenes, para determinar la influencia del incremento en la cantidad de datos con las relaciones de similitud que existen entre las palabras. El promedio de los porcentajes de similitud por colección de entrada se muestran en la figura 3.21 en la cual para ambos lenguajes

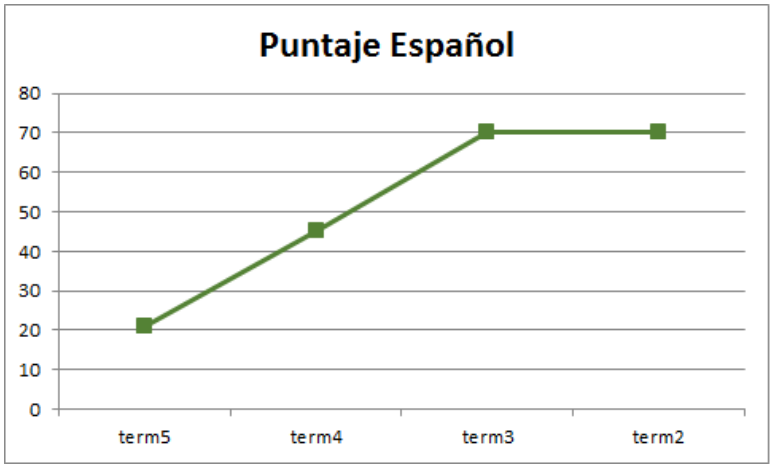
se da un incremento de los mismos al incrementar el tamaño de las colecciones.

Idioma	Term6	Term5	Term4	Term3	Term2	Term1	Term0
Inglés	20.00	78.66	76.00	84.00	94.66	90.66	105.33
Español	-	21.00	45.00	70.33	70.33	-	-

(a) Puntajes obtenidos por colección



(b) Gráfico de crecimiento de puntaje para pruebas en inglés

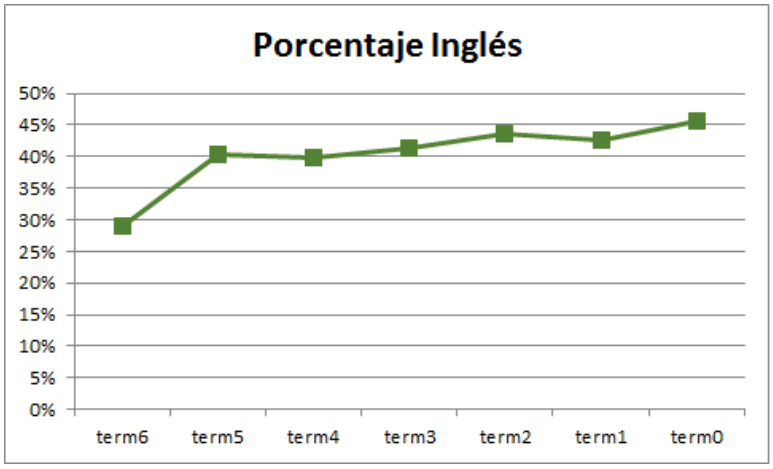


(c) Gráfico de crecimiento de puntaje para pruebas en español

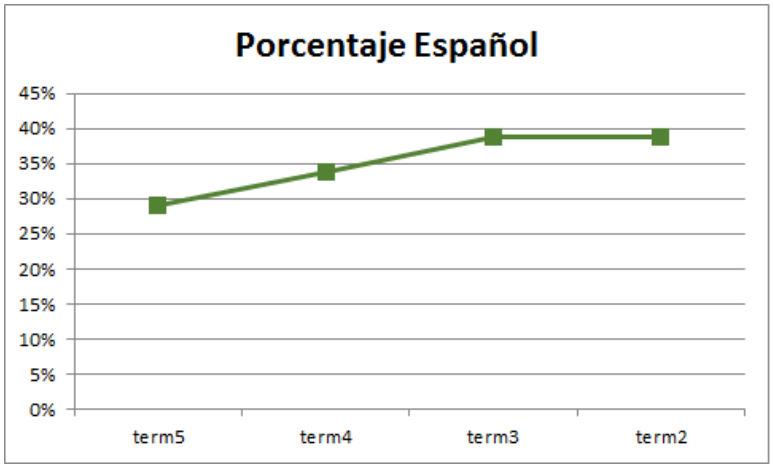
Figura 3.15: Puntajes obtenidos por idioma

Idioma	Term6	Term5	Term4	Term3	Term2	Term1	Term0
Inglés	28.9 %	40.3 %	39.8 %	41.4 %	43.4 %	42.7 %	45.5 %
Español	–	29.1 %	33.7 %	38.7 %	38.7 %	–	–

(a) Porcentajes de acierto obtenidos por colección



(b) Gráfico de crecimiento de porcentaje de acierto para pruebas en inglés



(c) Gráfico de crecimiento de porcentaje de acierto para pruebas en español

Figura 3.16: Porcentaje de acierto por idioma

	Term5	Term4	Term3	Term2	Term1	Term0
Term6	0,000	0,000	0,000	0,000	0,000	0,000
Term5	X	0,089*	0,092*	0,068*	0,075	0,031
Term4	X	X*	0,105*	0,039*	0,054	0,011
Term3	X	X*	X*	0,059*	0,077	0,017
Term2	X	X	X	X	0,125	0,059
Term1	X	X	X	X	X	0,030

(a) Porcentaje de nivel de significancia

	Term5	Term4	Term3	Term2	Term1	Term0
Term6	Sig	Sig	Sig	Sig	Sig	Sig
Term5	X	No Sig*	No Sig*	No Sig*	No Sig	Sig
Term4	X	X*	No Sig*	Sig*	No Sig	Sig
Term3	X	X*	X*	No Sig*	No Sig	Sig
Term2	X	X	X	X	No Sig	No Sig
Term1	X	X	X	X	X	Sig

Sig = Significativo, No sig = No Significativo

(b) Cambios significativos y no significativos

* Sección similar en tamaño a colecciones en español.

Figura 3.17: Sign Test: Inglés con todas las preguntas

	Term5	Term4	Term3	Term2	Term1	Term0
Term6	0,018	0,058	0,102	0,095	0,098	0,060
Term5	X	0,094*	0,045*	0,041*	0,035	0,076
Term4	X	X*	0,095*	0,080*	0,069	0,099
Term3	X	X*	X*	0,136*	0,122	0,087
Term2	X	X	X	X	0,149	0,081
Term1	X	X	X	X	X	0,061

(a) Porcentaje de nivel de significancia

	Term5	Term4	Term3	Term2	Term1	Term0
Term6	Sig	No Sig	No Sig	No Sig	No Sig	No Sig
Term5	X	No Sig*	Sig*	Sig*	Sig	No Sig
Term4	X	X*	No Sig*	No Sig*	No Sig	No Sig
Term3	X	X*	X*	No Sig*	No Sig	No Sig
Term2	X	X	X	X	No Sig	No Sig
Term1	X	X	X	X	X	No Sig

Sig = Significativo, No sig = No Significativo

(b) Cambios significativos y no significativos

* Sección similar en tamaño a colecciones en español.

Figura 3.18: Sign Test: Inglés únicamente con preguntas contestadas por todas las colecciones

	Term4	Term3	Term2
Term5	0,003	0,000	0,000
Term4	X	0,010	0,053
Term3	X	X	0,091

(a) Porcentaje de nivel de significancia

	Term4	Term3	Term2
Term5	Sig	Sig	Sig
Term4	X	Sig	No Sig
Term3	X	X	No Sig

Sig = Significativo, No sig = No Significativo

(b) Cambios significativos y no significativos

Figura 3.19: Sign Test: Español con todas las preguntas

	Term4	Term3	Term2
Term5	0,044	0,001	0,012
Term4	X	0,017	0,077
Term3	X	X	0,081

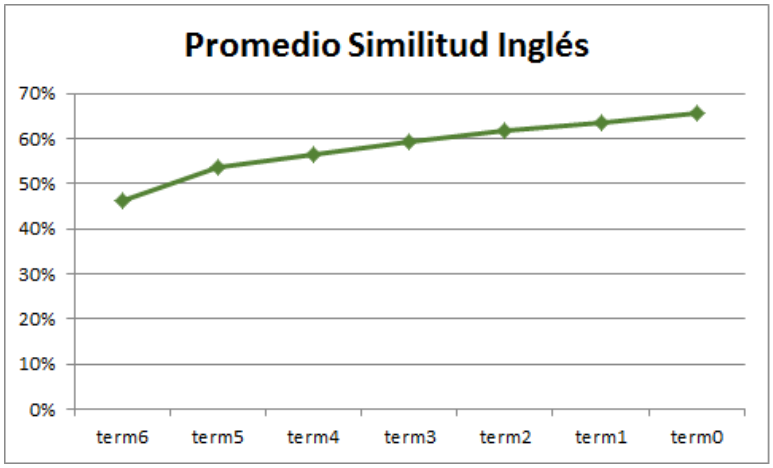
(a) Porcentaje de nivel de significancia

	Term4	Term3	Term2
Term5	Sig	Sig	Sig
Term4	X	Sig	No Sig
Term3	X	X	No Sig

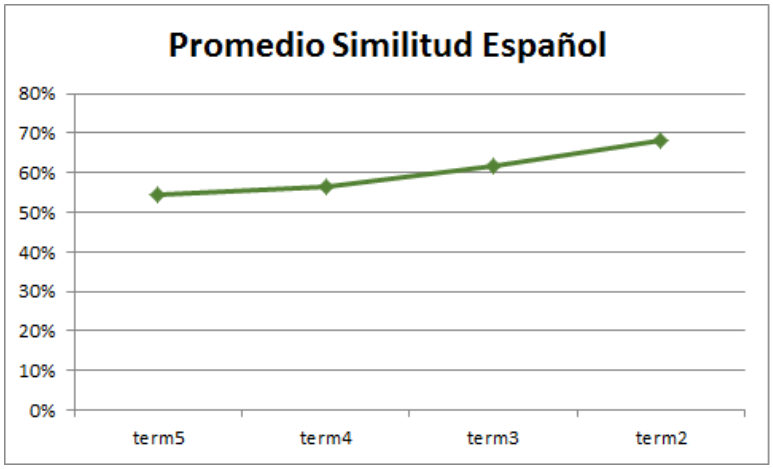
Sig = Significativo, No sig = No Significativo

(b) Cambios significativos y no significativos

Figura 3.20: Sign Test: Español únicamente con preguntas contestadas por todas las colecciones



(a) Gráfico de crecimiento de porcentajes promedio generales de similitud en inglés



(b) Gráfico de crecimiento de porcentajes promedio generales de similitud en español

Figura 3.21: Porcentajes promedio de similitud para la opción seleccionada

CAPÍTULO 4

Conclusiones y recomendaciones

There's no data like more data.

Banko and Brill

4.1 Conclusiones

El análisis comparativo entre los algoritmos SVD y SSVD determinó que debido, principalmente, a su capacidad de procesamiento distribuido, a su independencia con respecto a las dimensiones de las matrices de entrada, así como a las diferencias tan amplias en cuanto a los tiempos de ejecución, el algoritmo SSVD es superior a SVD Block Lanczos para trabajar con LSA. La precisión no fue un factor determinante para seleccionar SSVD sobre SVD, debido a que su precisión es comparable.

La capacidad para la detección de sinónimos del algoritmo LSA incrementa conforme se agregan más datos a la colección de entrada, ya que se observó un incremento en los resultados al evaluar preguntas de sinónimos de exámenes TOEFL y sus traducciones al

español en las colecciones cada vez más grandes; sin embargo, no en todos los casos el incremento fue estadísticamente significativo. Se observó un incremento estadísticamente significativo con respecto a las colecciones más pequeñas al incrementar la cantidad de datos principalmente debido a la introducción de una cantidad considerable de términos nuevos con cada incremento de la colección. En los casos de las colecciones más grandes, *Term2* a *Term0*, el incremento que se observa no es estadísticamente significativo ya que la cantidad de términos nuevos que se introducen, al manejar colecciones de un tamaño considerable de documentos, es mucho menor.

Se determinó que el algoritmo LSA para la detección de sinónimos es independiente del idioma en el que se está trabajando, debido a que el algoritmo trabaja con las relaciones latentes que existen entre los términos de una colección de documentos sin tomar en cuenta el significado o la semántica de los mismos. En los experimentos realizados se observó para las colecciones del mismo tamaño en inglés y español (*Term5* - *Term2*) que su comportamiento fue muy similar, sin embargo, para la colección en español no fue posible observar la mejora provista por la colección *Term0* de inglés. Al analizar además, el comportamiento de los porcentajes de similitud entre la palabra pregunta y la respuesta seleccionada se logró identificar un crecimiento similar en los experimentos en inglés y en español.

4.2 Recomendaciones y trabajo futuro

Trabajar con colecciones de entrada con datos "más limpios". Uno de los problemas observados con la colección de Wikipedia fue identificar y remover palabras no válidas del idioma con el cual se estaba trabajando. Sería posible utilizar un diccionario cuando se está creando la colección inicial para depurar los términos que formarán parte de la matriz. Esto ayudará a eliminar ruido en los datos, el cual influye en el contexto entre los diferentes términos.

Trabajar con colecciones de entrada temáticas para crear contextos más fuertes entre las palabras. Al utilizar una colección tan generalizada como Wikipedia, las palabras pasaban a formar parte de diversos contextos que se traslapaban, por ejemplo, palabras homónimas. Se observó además, como dependiendo de los documentos con los que se estuviera trabajando en un punto determinado, las palabras tenían una relación más fuerte con un contexto o con otro. Por ejemplo, se observa en la transición de las evaluaciones entre las colecciones *Term2* y *Term0* de la colección en inglés, como se da una pequeña disminución en *Term1* debido a la introducción de nuevos contextos.

Experimentar con *stemming* para determinar el nivel óptimo de transformación de las palabras que permita eliminar variantes irrelevantes como plurales y géneros, sin llegar a modificar tanto los términos que se invaliden los ejercicios de detección de sinónimos.

Probar la capacidad de este algoritmo para identificar los significados de acrónimos y siglas basados en la información de los documentos indexados. Para esto sería necesario la creación de una matriz más compleja donde un término podría estar formado por más de una palabra. En este caso la matriz crecería de forma muy acelerada, pero aun sería posible trabajar con LSA utilizando SSVD.

Bibliografía

[1] Apache mahout. http://mahout.apache.org/, oct 2012.

[2] Aspire introduction. https://wiki.searchtechnologies.com/index.php/Aspire_Introduction, nov 2012.

[3] Apache lucene - index file formats. http://lucene.apache.org/core/old_versioned_docs/versions/3_0_3/fileformats.html, oct 2012.

[4] Dimensional reduction. http://mahout.apache.org/users/dim-reduction/dimensional-reduction.html, 2012.

[5] Stochastic singular value decomposition. http://mahout.apache.org/users/dim-reduction/ssvd.html, 2012.

[6] Wikipedia. http://en.wikipedia.org/wiki/Wikipedia, nov 2012.

[7] David M. Blei, Andrew Y. Ng, and Michael I. Jordan. Latent dirichlet allocation. *J. Mach. Learn. Res.*, 3:993–1022, March 2003. ISSN 1532-4435. URL http://dl.acm.org/citation.cfm?id=944919.944937.

[8] Jeffrey Dean and Sanjay Ghemawat. Mapreduce: simplified data processing on large clusters. *Commun. ACM*, 51(1):107–113, January 2008.

[9] Scott Deerwester, Susan T. Dumais, George W. Furnas, Thomas K. Landauer, and Richard Harshman. Indexing by latent semantic analysis. *JOURNAL OF THE AMERICAN SOCIETY FOR INFORMATION SCIENCE*, 41(6):391–407, 1990.

[10] Chris H. Q. Ding. A probabilistic model for latent semantic indexing: Research articles. *J. Am. Soc. Inf. Sci. Technol.*, 56(6):597–608, April 2005. ISSN 1532-2882. doi: 10.1002/asi.v56:6. URL http://dx.doi.org/10.1002/asi.v56:6.

[11] Georges Dupret. Latent concepts and the number orthogonal factors in latent semantic analysis. In *Proceedings of the 26th annual international ACM SIGIR*

conference on Research and development in informaion retrieval, SIGIR '03, pages 221–226. ACM, 2003.

[12] EnglishTestStore. English synonyms tests. `http://englishteststore.net/index.php?option=com_content&view=article&id=81&Itemid=277`, 2008.

[13] George E. Forsythe, Michael A. Malcolm, and Cleve B. Moler. *Computer Methods for Mathematical Computations*. Prentice Hall Professional Technical Reference, 1977. ISBN 0131653326.

[14] John Gantz. The diverse and exploding digital universe. *International Data Corporation*, mar 2008.

[15] Gene H. Golub, Franklin T. Luk, and Michael L. Overton. A block lanczos method to compute the singular values and corresponding singular vectors of a matrix. Technical report, Stanford, CA, USA, 1977.

[16] Alon Halevy, Peter Norvig, and Fernando Pereira. The unreasonable effectiveness of data. *Communications of the ACM*, 24(2):8–12, 2009.

[17] Nathan Halko. *Randomized methods for computing low-rank approximations of matrices*. PhD thesis, University of Colorado at Boulder, 2012.

[18] A Kumaran, Ranbeer Makin, Vijay Pattisapu, Shaik Emran Sharif, Gary Kacmarcik, and Lucy V. Automatic extraction of synonymy information: An extended abstract.

[19] Chuck Lam. *Hadoop in Action*. Manning Publications Co., 12 2010.

[20] Jimmy Lin and Chris Dyer. Data-intensive text processing with mapreduce, 2010.

[21] Lin Loughheed. *Prentice Hall TOEFL Prep Book*. Prentice Hall, 1986. ISBN 0136966004.

[22] Michael McCandless., Erik Hatcher, and Otis Gospodnetic. *Lucene in Action*. Manning Pubs Co Series. Manning Publications, 2010.

[23] Tim Oates, Vinay Bhat, Vishal Shanbhag, and Charles Nicholas. Using latent semantic analysis to find different names for the same entity in free text. In *In Proceedings of the 4th international workshop on Web information and data management*, 2002.

[24] Mark D. Smucker, James Allan, and Ben Carterette. A comparison of statistical significance tests for information retrieval evaluation. In *Proceedings of the Sixteenth ACM Conference on Conference on Information and Knowledge Management*, CIKM '07, pages 623–632, New York, NY, USA, 2007. ACM. ISBN 978-1-59593-803-9. doi: 10.1145/1321440.1321528. URL `http://doi.acm.org/10.1145/1321440.1321528`.

[25] Donna Tatsuki. Basic 2000 words - synonym match 1. in: Interactive javascript quizzes for esl students. `http://www.aitech.ac.jp/~iteslj/quizzes/js/dt/mc-2000-01syn.html`, 1998.

[26] Peter Turney. Mining the web for synonyms: Pmi-ir versus lsa on toefl, 2001.

[27] Radim Řehůřek. Subspace tracking for latent semantic analysis. In *Proceedings of the 33rd European conference on Advances in information retrieval*, ECIR'11, pages 289–300, 2011.

[28] Tom White. *Hadoop: The Definitive Guide*. O'Reilly Media, 2012. ISBN 9781449338770.

[29] Anthony Zukas and Robert J. Price. Document categorization using latent semantic indexing. In *2003 Symposium on Document Image Understanding Technology*, pages 87–91, apr 2003.

Creación de Matriz de Entrada

1. Extraer la matriz de términos-documentos del índice de *Lucene*.

```
mahout lucene.vector --dir solr-4.3.0-example/solr/collection1/data/index --
    output noStemTermMatrix/term0-clean.vec --field text --idField id --dictOut
    dict/nostem_term0-clean.txt --norm 2 --maxPercentErrorDocs 1
```

2. Colocar el diccionario términos en el Sistema de Archivos Distribuido de Hadoop.

```
hadoop fs -put dict/nostem_term0-clean.txt dict/nostem_term0-clean.txt
```

3. Remover de la matriz los términos de muy baja frecuencia. 10 en este caso.

```
hadoop jar mahout-cmds/remove-one-frequency-1.0-jar-with-dependencies.jar tec.
    ralfaro.hadoop.RemoveMRFrequency noStemTermMatrix/term0-clean.vec
    cleanNoStemTermMatrix/term0-clean -remote dict/nostem_term0-clean.txt
    cleanDict/nostem_term0-clean.txt 10
```

4. Calcular la descomposición de valores singulares por medio de SSVD con una dimensionalidad espacial de 200. La matriz V es la matriz ortogonal de términos y la matriz U es la matriz ortogonal de documentos.

```
mahout ssvd --input cleanNoStemTermMatrix/term0-clean/part-r-00000 --output
    ssvd200CleanTermMatrixQ1/term0 --rank 200 --reduceTasks 1 --tempDir /tmp/
    ssvd200term0/ --vHalfSigma false --uHalfSigma false --computeV true --
    computeU true --powerIter 1
```

5. Transponer la matriz V multiplicarla con la matriz diagonal S.

```
mahout transpose --input ssvd200CleanTermMatrixQ1/term0/V --tempDir /tmp/
    ssvd200clean/transpose --numRows 627307 --numCols 200
```

```
hadoop fs -mv ssvd200CleanTermMatrixQ1/term0/transpose* ssvd200CleanTermMatrixQ1/
    term0/V_trans
```

6. Multiplicación de la matriz diagonal S por la matriz transpuesta V.

```
hadoop jar mahout-cmds/sigma-multiplication-1.0-jar-with-dependencies.jar tec.
    ralfaro.hadoop.SigmaMultiplication ssvd200CleanTermMatrixQ1/term0/sigma
    ssvd200CleanTermMatrixQ1/term0/V_trans ssvd200NoStemOutput/term0-clean
```

7. Transponer el resultado de la multiplicación de SV' para obtener la matriz entrante para la evaluación de sinónimos.

```
mahout transpose --input ssvd200NoStemOutput/term0-clean --tempDir /tmp/
    ssvd200NoStemOutput/term0-clean/transpose --numRows 200 --numCols 627307
```

```
hadoop fs -mv ssvd200NoStemOutput/transpose* ssvd200NoStemOutput/term0-clean-for-
    synonyms
```

Término	S0	S1	S2	S3	...	Sn
caballo	0.54	0.34	0.45	0.12	...	0.05
computadora	0.64	0.24	0.05	0.17	...	0.08
pato	0.24	0.12	0.41	0.22	...	0.15
termino	0.12	0.61	0.15	0.32	...	0.03
...	...	...	...	...	...	...

Cuadro 1: Ejemplo de SV' con pesos basados en los TF-IDF originales para los n valores más significativos de S

Evaluación de un examen de sinónimos

Para la evaluación de exámenes de sinónimos se implementó una aplicación de *Hadoop* que ejecuta subrutinas de *MapReduce* para la evaluación individual de cada pregunta del examen.

La aplicación recibe como parámetros de entrada lo siguiente:

- Examen a evaluar (colección de evaluación)
- Ubicación de los resultados.
- Diccionario de términos disponibles en la colección de entrada
- Matriz TS (colección de entrada)
- Ubicación en *Hadoop* para guardar las listas de similitud
- Si se desea aplicar o no *stemming* (para todas nuestras pruebas, no se aplicó *stemming*)

```
hadoop jar evaluate-synonym-exam-1.0-jar-with-dependencies.jar tec.ralfaro.hadoop.
    EvaluateSynonymExam exam/questions/exam3.txt exam/answers/exam3/exam3-term1.txt
    dict/nostem_term1-clean.txt ssvd200/term1-clean-for-synonyms answers3/
    synonymQuestionEval200Term1 false
```

Por cada pregunta, se genera un registro de salida en un archivo separado por tabulaciones. Cada registro contiene la siguiente información:

- Pregunta
- Respuesta Correcta
- Indicador de si la respuesta seleccionada como correcta realmente la correcta
- Opción seleccionada como correcta
- Posición en la lista de la opción seleccionada como correcta
- Valor de similitud de la opción seleccionada como correcta
- Opción ordenada como segunda

- Posición en la lista de la ordenada como segunda
- Valor de similitud de la ordenada como segunda
- Opción ordenada como tercera
- Posición en la lista de la ordenada como tercera
- Valor de similitud de la ordenada como tercera
- Opción ordenada como cuarta
- Posición en la lista de la ordenada como cuarta
- Valor de similitud de la ordenada como cuarta

Colecciones de evaluación: Exámenes de prueba

.1 Español

.1.1 Examen 1

Pregunta	Opción Correcta	Opción 2	Opción 3	Opción 4
oxidado	corroido	negro	sucio	pintado
laton	metal	madera	piedra	plastico
vuelta	giro	dolor	sudor	rubor
pasaje	pasillo	tiquete	entrada	salon
dar	presentar	desafiar	jactar	despreciar
apoyar	descansar	raspar	rallar	consultar
barril	tonel	botella	caja	estuche
molestia	peste	basura	alivio	molesto
manta	alfombra	sofa	otomano	pasillo
grifo	desagüe	hervir	golpear	rap
dividido	separado	aplastado	rallado	magullado
terron	pedazo	tallo	tronco	extremidades
bosquejo	contorno	par	mezcla	bloque
jurar	prometer	explicar	pensar	describir
aliviado	descansado	somnoliento	cansado	apresurado
ameritar	merito	necesidad	quiera	esperar
prisa	apresuramiento	ira	escuchador	rencor
rigido	firme	oscuro	borracho	cocinado
verso	seccion	hierba	ramita	rama
empinado	escarpado	desnudo	robusto	piedra
envidioso	celoso	entusiasta	dolor	alivio
pega	pasta	jarabe	bloque	jalea
despreciar	rechazar	disfrutar	evitar	planificar
referir	orientar	llamar	realizar	explicar
miembro	rama	corteza	tronco	ramita
almohadilla	almohadon	tabla	bloque	tableta
presumir	alardear	gritar	quejarse	explicar
aplaudir	aprobacion	miedo	vergüenza	amigos
pagina	hoja	libro	bloque	grifo
vastago	tallo	corteza	columna	tronco
agarrar	tomar	consultar	pedir	ceder
maletero	baul	bolsa	closet	columpio
yerbajo	semillero	ropa	animal	vegetal
aprobacion	aceptacion	regalo	declaracion	confesion
masa	bulto	servicio	culto	elemento
columpio	balanceo	rebote	salto	caida
dolor	doloroso	rojo	caliente	aspero
obstaculizar	bloquear	ayudar	aliviar	ceder
pegajoso	viscoso	suave	brillante	humedo
confesion	declaracion	servicio	culpabilidad	convenio
tejer	entrelazar	imprimir	estampar	sacudir
platillo	plato	caja	disco	tarro
sustancia	materia	postura	nivel	puntuacion
firmemente	tenazmente	renuente	desgraciadamente	esperanzadamente
retorcer	entrelazar	recortar	sujetar	rizar
rascar	rallar	cortar	picar	rebanar
moler	machacar	rebanar	golpear	golpear
aumentar	agrandar	mover	rizar	sacudir
cosecha	ingesta	tallo	bulto	dividir
aprobar	apoyar	alardear	despreciar	enojar

Cuadro 2: Examen de evaluación 1

.1.2 Examen 2

Pregunta	Opción Correcta	Opción 2	Opción 3	Opción 4
implicar	sugerir	apunalar	grueso	destruir
deambular	divagar	nudo	confundir	maravilla
beneficioso	ventajoso	ayuda	caridad	sabio
flama	fogata	judicial	temperamento	estilo
negligentes	descuidados	pijamas	morbido	oscuro
distante	reservado	encima	ordenado	inteligente
resolver	determinar	turno	rompecabezas	querer
congregar	reunir	adoracion	molestar	prisa
pronunciar	expresar	aplazar	prestar	reir
intrepido	valiente	poderoso	cobarde	cuidadoso
despreciable	insignificante	discutible	descuidado	oscuro
placido	tranquilo	lento	solemne	desviado
rastrillo	horquilla	delgado	caballero	pala
engano	truco	ranchero	simplon	droga
estigma	mancha	juicio	dificultad	santidad
residir	morar	permanecer	hogar	sedimentos
codicioso	avaro	tranquilo	disimulado	acecho
respetar	tolerar	odiar	asistir	vivo
sagaz	astuto	intratable	media	inteligente
estorba	obstaculiza	rancio	alabanza	persisten
remoto	distante	automatico	salvaje	media
detestar	odiar	discutir	descubrir	revelar
amable	agradable	bonito	inteligente	presente
predecir	pronosticar	determinar	prevenir	descubrir
familiares	parientes	regocijan	giro	amigo
pensativo	meditabundo	oprimido	enjaulado	feliz
destierro	exilio	odio	desvanecimiento	limpio
fraude	engano	descontento	impostor	payaso
sacarina	dulce	deje	arido	dejar
arrastrar	jalar	somnoliento	aplaste	orgulloso
enojo	ira	nudo	crimen	humo
pletora	riqueza	problemas	tonto	amor
calamidad	desastre	pocion	silencio	mariscos
pomposo	arrogante	apoyo	ocupado	llamativo
frecuente	prevalente	viento	servil	raro
respingo	estremecimiento	alegria	aplaste	solitario
superficial	llano	hermosa	inteligente	rico
enredo	enmaranar	grunido	danza	escalofrio
reformar	corregir	castigar	destruir	desplegar
metodico	sistematico	ritmico	poetico	irrespetuoso
flagrante	deslumbrante	vibrante	vicioso	agradable
mitigar	aliviar	mezclar	defender	confundir
sancionar	reganar	empujar	probar	gritar
fusionar	mezclar	caracter	distincion	firmeza
mitin	asamblea	demostrar	supuesto	verdad
abyecto	abatido	indigente	deseo	extremo
medido	ajustado	amable	tranquilo	guapo
timido	cauteloso	apatico	arrogante	pendenciero
propuesta	oferta	mendicante	gandul	predecir
mordaz	caustico	muertos	horrible	fetido
grosero	tosco	infantil	tempestuosa	desordenado
avaro	mezquino	bondadoso	picante	sureno
herir	atacar	huir	escapar	suciedad
aventar	lanzar	tallar	enrollar	llevar
agotar	vaciar	decorar	mendigar	precisar
voluntario	complaciente	caridad	preso	descuidado
rechazar	negar	basura	ofrecer	dificil
enganar	defraudar	tacano	argumentar	pecar
desdichado	infeliz	cruel	equivocado	extranar
antiguo	clasico	alcoholico	enfermo	estropeado
agrio	acido	derecho	enojado	deseable
atrapar	acorralar	desplegar	pintar	acelerar
animo	entusiasmo	relleno	preocupacion	juicio
regatear	negociar	cansancio	ascender	descender
impulsar	forzar	bloquear	obstaculizar	desacreditar
multitud	muchedumbre	ropa	campana	peso
imperial	real	malcriado	opresivo	hermoso
difuso	disperso	dificil	incomprensible	desconectado
dificultar	entorpecer	comprobar	desatar	perder
latente	inactivo	reciente	afeminado	deseable
desventurado	miserable	torcido	forzado	aumentado
fastidioso	molesto	indignante	temible	empobrecido
regular	supervisar	nivelar	fluir	posicion
probar	justificar	excavar	ocultar	integridad
prolongar	demorar	contratar	doblar	acorralar
flojo	holgado	seguro	sirviente	esforzado
rigor	austeridad	cuerda	arreglo	exceso
diferente	distinta	salir	disminuir	despilfarrar
agil	flexible	lento	honesto	aburrido
jovial	alegre	incredulo	repugnante	mareado
indiferente	neutral	cruel	precioso	promedio
simular	imitar	excitar	enganar	gritar
carisma	encanto	fantasma	fuerza	coraje
porcionar	dividir	decidir	cortar	disputar
generico	general	barato	fresco	anciano
escrupulo	remordimiento	angustia	impunidad	perseverancia
cauteloso	prudente	calmado	curva	confundido
distorsionado	deforme	equivocado	malvado	perjuicio
suntuoso	lujoso	delirante	magnifico	peligroso
rico	millonario	dudoso	fiesta	inverso
indecoroso	indecente	desmanado	fino	brillante

Pregunta	Opción Correcta	Opción 2	Opción 3	Opción 4
saciar	llenar	iluminar	cargar	horror
estentorea	fuerte	violenta	bastardo	sigiloso
sacrosanta	sagrado	oracion	santuario	piadosa
modesto	humilde	atractivo	inteligente	corriente
costumbre	habito	postre	etica	entrega
extender	prolongar	inquirir	relajar	esperar
prisa	apuro	danza	ocupado	limpio
solemne	serio	divertido	danino	discurso
carrete	bobina	pescado	golpe	error
pandilla	club	varios	coqueteo	socializar
nefasto	malo	infame	macabro	distinguido
enchilado	picante	despilfarro	juego	plano
sobrenatural	extraordinario	inmaduro	eliminado	ganado
pernicioso	nocivo	ilicita	abierto	indeciso
represalia	venganza	acusacion	detestan	insinuacion
multiple	diverso	evidente	contemporaneo	dispuestos
jocoso	gracioso	farsa	fiable	argumentativo
servidumbre	trabajo	estupidez	maldad	investigacion
prerrequisito	necesario	supuesto	dificil	alquitran
grave	urgente	cuestionable	franco	traidor
aferrarse	luchar	atrapar	reirse	intencion
diversos	varios	edades	suministro	tremenda
sustituir	reemplazar	crecer	deshacer	preguntar
venerar	reverenciar	ordenar	respirar	pulir
conciliar	apaciguar	desacuerdo	revivir	separar
exultante	jubiloso	miedo	expectante	exigiendo
subrepticia	clandestino	prepotente	indirectos	impia
rencor	despecho	alegria	bebida	maravilla
escala	asciende	vela	natacion	patina
borroso	desenfoque	regodearse	residuos	celebrar
llovizna	aspersion	rizado	dorar	empape
mundano	comun	sucio	confuso	extraordinaria
pretension	ambicion	estres	residuos	luchas
influir	cultivar	resultado	compartir	pomposa
heraldo	anuncio	insignias	posponer	real
facultad	profesorado	defectuoso	escuela	deseo
regocijo	alegria	ira	sarcasmo	desconfianza
menospreciar	despreciar	devaluar	erronea	codiciar
impugnar	invadar	implicar	traste	recalcitrante
sobrevenir	seguir	intervenir	extralimitacion	desplazamiento
exigente	urgente	tratado	avaro	expediente
ferviente	ardiente	encantadora	dificil	obstinado
sustituto	artificial	caotica	improvisada	vaga
fragante	oloroso	ubicuo	timido	belicosa
vociferante	estridente	numeroso	abundante	charlatan
vileza	depravacion	letargo	honor	beligerancia
homogenea	igual	extrano	cortes	alcalina
concentrador	centro	abogado	anciano	extension
manso	sumiso	perdido	evasiva	agresivo
fastidiar	irritar	encogerse	devorar	evitar
aparecer	emerger	desaparecer	cortar	ensenar
irregular	erratico	enojado	cansado	pronuncia
chillon	llamativo	masiva	consciente	gustoso
alarde	desfile	ocultamiento	confianza	refriega
flexible	doblar	borrachera	librado	consumir
tentar	ligar	pronunciar	reinar	igual
vil	cobarde	devastadora	inteligente	generoso
aficionado	novato	embaucador	devoto	agente
contigua	adjunto	captura	dividido	circunstancial
estafador	timador	experto	divinidad	elegante
picaro	bribon	pasear	bufon	color
apologista	defensor	mentiroso	fracaso	admirador
intermediario	delegado	espasmo	cercania	corte
abofetear	golpear	proteger	barricada	armario
parodia	burla	confusion	desastre	especulacion
enojado	furioso	distante	saludar	dudar
amonestar	advertir	alabar	decorar	admirar
pedir	suplicar	retirar	enganar	preguntar
aplomo	equilibrio	mina	torpeza	complicacion
afirmar	declarar	disipar	crear	odiar
transporte	carruaje	promedio	vicioso	disfraz
encajar	ajustar	enigma	contraccion	pesar
auspicio	patrocinio	supersticion	referencia	archivo
sepultura	entierro	parasito	verso	sermon
arpia	bruja	gobernador	buque	abogado
apotegma	adagio	medicina	especulacion	resistencia
grandilocuencia	fanfarroneria	respeto	negacion	solemnidad
fulminacion	explosion	recesion	logro	bendicion
hastiado	displicente	nativo	caliente	oculto
acantilado	precipicio	advertencia	camping	tumba
plutocrata	banquero	sacerdote	juez	astronomo
inescrutable	misterioso	dificil	inflexible	cauteloso
afliccion	consternacion	delirante	cubierto	confusion
integro	honorable	horizontal	humilde	flojo
ensueno	fantasia	fantasma	curiosidad	pesadilla
botin	despojos	destruccion	residuos	desastre
locuaz	hablador	sediento	hermoso	complicado
quimera	ilusion	chimenea	protesta	estilo
temeridad	audacia	temor	timidez	estupidez
deducir	inferir	demandar	ideal	ilegal
pedante	pretencioso	arduo	voluble	consecuente

Pregunta	Opción Correcta	Opción 2	Opción 3	Opción 4
engreido	arrogante	atrasado	torpe	accidentado
penitencia	sacrificio	respiracion	inmigracion	divergencia
abominable	villano	deliberada	fatigante	habitual
seducir	persuadir	complejidad	enganar	angustiar
atento	considerado	migracion	consolidado	destinado
divulgar	difundir	puntuar	forzar	disuadir
elevar	levantar	disminuir	mencionar	colocar
nombrar	designar	calificar	discutir	garantizar
acumular	almacenar	excavar	mio	postre
verdadero	autentico	profundo	antiguo	irascible
absoluto	completo	audaz	convincente	denso
naturaleza	esencia	volumen	cumbre	liberacion
destreza	habilidad	moreton	recuerdo	sinverguenza
adecuado	oportuno	inesperada	hablado	idea
resplandor	brillo	ocultar	mostrar	convocar
erratico	irregular	atractiva	frecuente	dificil
civil	publico	rudo	trillado	cuestionable
pares	companeros	manzana	connotan	recluso
fiasco	desastre	festival	casualidad	ceremonia
abismo	barranco	encanto	freno	criticar
especializacion	maestria	actividad	valor	esfuerzo
estrafalario	absurdo	distante	pastoral	beligerante
implorar	pedir	limpiar	odiar	resolver
proeza	hazana	respuesta	accidente	persuadir
reacio	rebelde	vacilante	sometido	reste
bonita	guapa	llano	confuso	fea
edicto	decreto	desalojar	correccion	destino
enaltecer	alabar	impuesto	gravar	reganar
expectar	esperar	equivocar	repugnante	novato
sentencia	peaje	bosquecillo	colina	sonajero
somnifero	hipnotico	juvenil	borracho	encantador
iterar	repetir	inestable	empobrecer	anunciar
baluarte	muralla	enigma	festival	confuso
culminacion	realizacion	desastres	casualidad	persuasion
aparentar	fingir	empujon	desmayo	miedo
propicio	favorable	enganosa	presentimiento	peligroso
tactica	truco	fiesta	testimonio	oracion
voraz	hambriento	violento	voluble	bravucon
facil	simplista	capacidad	seccion	vengativo
evitar	abstener	revertir	acompanar	admirar
fugar	huir	robar	oscura	absolver

Cuadro 3: Examen de evaluación 2

.1.3 Examen 3

Pregunta	Opción Correcta	Opción 2	Opción 3	Opción 4
apreciar	respetar	acuerdo	contradiccion	entender
abogar	recomendar	insistir	apreciar	deplorar
calificaciones	credenciales	medidas	formas	vestimenta
retraer	retirar	denegar	disculpar	corregir
revelar	difundir	explicar	descubrir	reconocer
predecir	pronosticar	seguro	miedo	explicar
cantidades	montos	articulos	variedades	placas
reducida	disminuida	implementado	aumentada	reevaluado
fabricado	construido	garantizado	localizado	asegurado
aproximadamente	cercanamente	esencialmente	confidencialmente	sinceramente
recreacion	diversion	necesidades	educacion	imprevistos
recientemente	ultimamente	cualquiera	dentro	fuera
preparados	listos	ansiosos	ida	asustados
contemporaneo	concurrente	antiguo	moderno	varios
simboliza	representa	resume	revela	contiene
socios	colegas	vecinos	superiores	estudiantes
descubrimiento	hallazgo	eleccion	distrito	danza
negociado	arreglado	rechazados	sugerido	exigido
concluyente	definitivo	comprendido	valido	predicho
censo	encuesta	grupo	numero	periodico
ordinario	comun	inusual	noticioso	publico
fluctuado	variado	rosa	estabilizado	disminuido
controversial	disputable	popular	personal	sesgada
compasion	misericordia	desconfianza	repulsion	conocimiento
cansancio	agotamiento	ansia	acaparamiento	mantenimiento
agresivo	violento	deprimido	inestable	inseguro
dedicado	devoto	cortes	agradable	considerable
demostrar	mostrar	adivinar	innecesario	describir
seguridad	garantia	dependencia	aprobacion	presencia
extingir	matar	recuerdar	enchufe	inicio
extendido	ampliado	envio	pasado	anuncio
practicamente	casi	basicament	siempre	convenientemente
inepto	inapropiado	eficiente	inteligente	inferior
aludido	referido	falsificado	olvido	relato
celoso	ardiente	colorido	grosero	listo
enemistad	odio	relacion	amistad	cercania
detallado	redundante	interesante	conciso	inteligente
intransigente	terco	honesto	amable	fuerte
letargico	indiferente	sed	calor	descuidada
salvaje	feroz	amistoso	peludo	independiente
innato	natural	oculto	benevolente	altruista

Pregunta	Opción Correcta	Opción 2	Opción 3	Opción 4
distincion	diferencia	atraccion	peligro	comodidad
fundamental	basico	permanente	interesante	antiguo
indica	sugiere	admite	insiste	predica
brillante	inteligente	conocido	profesional	popular
exactamente	precisamente	intuitivamente	inmediatamente	brevemente
descortes	grosero	guapo	irritable	perdido
excluye	prohibe	evita	rie	invita
sentido	punto	final	longitud	titulo
desaparecer	desvanecer	comportar	olvidar	feliz
sintetico	artificial	caro	moda	nuevo
riesgoso	peligroso	dificil	resbaladizo	emocionante
tranquilo	pacifico	mareado	triste	sonoliento
introvertido	reservado	termino	odioso	precauciones
asesinato	homicidio	honrado	elegido	politico
expansion	ampliacion	problemas	pintura	disminucion
huyo	corrio	cayo	vino	tardo
audible	escuchado	visto	sentido	ignorado
esencial	integral	posible	menor	negativo
horrible	espantoso	encantador	espectacular	obsceno
abiertamente	publicamente	dificilmente	ofensivamente	extremadamente
secreto	reservado	inusual	somnolencia	enfermizo
afin	relacionado	amable	cortes	jocososo
ridiculo	comico	fuerte	sombrio	comun
matices	sutilezas	imagenes	ritmo	rimas
oportuno	adecuado	esperado	afortunado	anticipado
tentar	seducir	disuadir	informar	enganar
destruido	borrado	cubierto	producido	indicado
sellado	cerrado	oculto	inalcanzable	antiguo
reproducido	impreso	tomados	renovado	restaurado
inaccesible	impenetrable	solitario	alto	seco
imitar	copiar	criticar	admirar	recuerde
concedido	asumido	esperado	garantizado	convencido
cautivado	encantado	asustado	aburrido	disgustado
precede	antes	punto	indica	causa
severidad	seriedad	proposito	ubicacion	autor
recoger	acumular	enterrar	desear	requerir
fiable	confiable	automatico	versatil	rapido
tipico	caracteristico	completo	total	duro
desacostumbrado	desutilizado	sorprendido	decepcionado	miedo
avergonzado	humillado	infeliz	decepcionado	asqueado
inapropiada	inadecuada	innecesaria	incoherente	incapaz
convincente	persuasivo	realista	fiable	claro
adecuado	correcto	legal	suficiente	positiva
manso	docil	solo	perezoso	ruidoso
falso	simulado	caro	robado	vendido
potencial	posible	cierto	real	definitivo
desplazado	apartado	dirigido	apresurado	devuelto
primordialmente	principalmente	objetivamente	activamente	subjetivamente
conflicto	desacuerdo	decision	condicion	accion
pausa	intervalo	introduccion	oracion	juego
energico	animado	corto	largo	solemne
torpe	inepto	superior	ignorado	notado
eliminado	borrado	corregido	circulo	determinado
feroz	salvaje	peludo	tonto	insensible
acosar	molestar	preguntar	sobornar	persuadir
prendas	ropas	casas	fotografias	herramientas
incompatible	antagonico	distinto	diferente	incomparable
apagar	extinguir	crear	prolongar	desalentar
sospechoso	acusado	incredulo	juez	robo
pintoresco	curioso	antiguo	elaborado	ultramoderna
masivo	enorme	ligero	repetitivo	infrecuente
disposiciones	especificaciones	escritores	lectores	abogados
renunciar	abandonar	mantenga	controlar	ganar
disonancia	melodia	claridad	discordia	volumen
asunto	tema	libro	articulo	ecuacion
satisfactorio	adecuado	caprichoso	audaz	comprensivo
donar	dar	suministrar	ofrecer	mostrar
maduro	crecido	inteligente	serio	infantil
ensayar	practicar	cantar	escribir	introducir
reanudo	continuo	regreso	repitio	retrocedio
prejuicio	parcialidad	ira	groseria	accion

Cuadro 4: Examen de evaluación 3

.2 Inglés

.2.1 Examen 1

Pregunta	Opción Correcta	Opción 2	Opción 3	Opción 4
rusty	corroded	black	dirty	painted
brass	metal	wood	stone	plastic
spin	twirl	ache	sweat	flush
passage	hallway	ticket	entrance	room
yield	submit	challenge	boast	scorn

Pregunta	Opción Correcta	Opción 2	Opción 3	Opción 4
lean	rest	scrape	grate	refer
barrel	cask	bottle	box	case
nuisance	pest	garbage	relief	troublesome
rug	carpet	sofa	ottoman	hallway
tap	drain	boil	knock	rap
split	divided	crushed	grated	bruised
lump	chunk	stem	trunk	limb
outline	contour	pair	blend	block
swear	vow	explain	think	describe
relieved	rested	sleepy	tired	hasty
deserve	merit	need	want	expect
haste	hurry	anger	ear	spite
stiff	firm	dark	drunk	cooked
verse	section	weed	twig	branch
steep	sheer	bare	rugged	stone
envious	jealous	enthusiastic	hurt	relieved
paste	dough	syrup	block	jelly
scorn	refuse	enjoy	avoid	plan
refer	direct	call	carry	explain
limb	branch	bark	trunk	twig
pad	cushion	board	block	tablet
boast	brag	yell	complain	explain
applause	approval	fear	shame	friends
sheet	leaf	book	block	tap
stem	stalk	bark	column	trunk
seize	take	refer	request	yield
trunk	chest	bag	closet	swing
weed	seedling	cloth	animal	vegetable
approval	endorsement	gift	statement	confession
mass	lump	service	worship	element
swing	sway	bounce	break	crash
sore	painful	red	hot	rough
hinder	block	assist	relieve	yield
sticky	gooey	smooth	shiny	wet
confession	statement	service	plea	bargain
weave	intertwine	print	stamp	shake
saucer	dish	box	frisbee	can
substance	thing	posture	level	score
firmly	steadfastly	reluctantly	sadly	hopefully
twist	intertwine	clip	fasten	curl
scrape	grate	chop	mince	slice
grind	rub	slice	hit	tap
swell	enlarge	move	curl	shake
harvest	intake	stem	lump	split
approve	support	boast	scorn	anger

Cuadro 5: Examen de evaluación 1

.2.2 Examen 2

Pregunta	Opción Correcta	Opción 2	Opción 3	Opción 4
imply	suggest	stab	thick	destroy
ramble	wander	knot	confuse	wonder
beneficial	advantageous	help	charity	wise
flare	blaze	judicial	temper	style
negligent	careless	pajamas	morbid	dark
aloof	reserved	above	tidy	clever
resolve	decide	turn	puzzle	want
congregate	gather	worship	disturb	hurry
utter	express	defer	borrow	laugh
fearless	brave	powerful	cowardly	careful
negligible	insignificant	arguable	careless	dark
placid	calm	lazy	solemn	devious
rake	scoundrel	thin	gentleman	shovel
dupe	trick	rancher	simpleton	drug
stigma	stain	trial	difficulty	holiness
reside	dwell	remain	home	sediment
covetous	greedy	quiet	sneaky	lurking
abide	endure	hate	attendance	live
shrewd	astute	intractable	mean	intelligent
fetter	hamper	rancid	praise	persist
remote	distant	automatic	savage	mean
detest	hate	argue	discover	reveal
gracious	pleasant	pretty	clever	present
predict	foretell	decide	prevent	discover
kin	relative	exult	twist	friend
pensive	thoughtful	oppressed	caged	happy
banish	exile	hate	fade	clean
fraud	argument	malcontent	imposter	clown
saccharine	sweet	leave	arid	quit
drag	pull	sleepy	crush	proud
wrath	anger	knot	crime	smoke
plethora	wealth	trouble	foolish	love
calamity	disaster	potion	silence	shellfish
pompous	arrogant	supportive	busy	gaudy
prevalent	widespread	wind	servile	rare
wince	flinch	cheer	crush	solitary

Pregunta	Opción Correcta	Opción 2	Opción 3	Opción 4
superficial	shallow	gorgeous	intelligent	rich
tangle	snarl	growl	dance	shiver
reform	correct	punish	destroy	display
methodical	systematic	rhythmic	poetic	disrespectful
flagrant	glaring	vibrant	vicious	pleasant
mitigate	relieve	blend	defend	confuse
rail	scold	push	try	punish
meld	blend	character	distinction	firmness
rally	muster	demonstrate	course	truly
abject	despondent	indigent	desire	extreme
bespoke	tailored	gentle	quiet	handsome
diffident	shy	apathetic	arrogant	quarrelsome
proffer	tender	mendicant	wastrel	predict
mordant	caustic	dead	gruesome	fetid
churlish	boorish	childish	tempestuous	disorderly
picayune	paltry	petty	spicy	southern
smite	strike	flee	speck	dirt
winnow	weed	carve	wind	carry
deplete	exhaust	decorate	beg	hurry
voluntary	willing	charity	prisoner	careless
refute	garbage	deny	offer	difficult
cheat	defraud	stingy	argue	freckle
miserable	unhappy	cruel	wrong	miss
vintage	classic	alcoholic	disease	spoiled
tart	acid	law	angry	desirable
corner	trap	display	paint	hurry
zest	gusto	cram	worry	trial
haggle	bargain	tired	climb	decrease
impel	force	block	hinder	discredit
throng	mass	garment	bell	weight
imperial	regal	bratty	oppressive	beautiful
diffuse	scatter	difficult	incomprehensible	unplug
hinder	check	lose	loose	despair
latent	dormant	recent	effeminate	desirable
wretched	miserable	twisted	forced	increased
irksome	annoying	outrageous	fearsome	impoverished
regulate	police	even	flow	position
warrant	justify	burrow	hide	integrity
protract	delay	hire	fold	corner
lax	slack	ensure	servant	strive
rigor	austerity	rope	fix	excess
discrete	distinct	leave	diminish	squander
lissome	supple	slow	honest	dull
jovial	merry	incredulous	revolting	dizzy
indifferent	neutral	unkind	precious	mean
simulate	imitate	excite	trick	apelike
charisma	charm	ghost	force	courage
apportion	divide	decide	cut	squabble
generic	general	cheap	fresh	elderly
qualm	scruple	distress	impunity	persevere
wary	cautious	calm	curved	confused
distort	deform	wrong	evil	harm
sumptuous	luxurious	delirious	gorgeous	perilous
nabob	bigwig	doubter	frolic	converse
louche	indecent	gauche	fine	brilliant
pall	satiate	light	carry	horror
stentorian	loud	violent	misbegotten	stealthy
sacrosanct	sacred	prayer	sanctuary	pious
modest	humble	attractive	clever	current
custom	habit	dessert	ethic	deliver
prolong	extend	inquire	relax	wait
hustle	hurry	dance	busy	clean
solemn	serious	amusing	harmful	speech
reel	whirl	fish	hit	mistake
inscrutable	mysterious	difficult	inflexible	wary
appall	dismay	delirious	covered	confuse
upright	honorable	horizontal	humble	supine
reverie	daydream	palimpsest	phantom	curio
loot	spoils	destruction	waste	cavort
loquacious	talkative	thirsty	beautiful	complicated
chimera	illusion	chimney	protest	panache
temerity	audacity	fearfulness	shyness	stupidity
educe	elicit	demand	ideal	unlawful
pedantic	pedestrian	arduous	fickle	consequential
bumptious	arrogant	backward	clumsy	rugged
expiation	atonement	breathing	immigration	divergence
flagitious	villainous	deliberate	fatiguing	habitual
inveigle	cajole	complexity	hoodwink	distress
heed	consider	trek	consolidate	bound
edge	diffuse	point	force	dissuade
elevate	hoist	lessen	mention	affix
appoint	nominate	score	discuss	ensure
hoard	stockpile	burrow	mine	dessert
veritable	authentic	deep	ancient	irascible
unmitigated	utter	audacious	unpersuasive	dense
epitome	essence	volume	summit	deliverance
edict	decree	vacate	correction	destiny
extol	praise	tax	burden	berate
abeyant	pending	False	disgusting	novice
knell	toll	copse	hill	rattle
soporific	hypnotic	juvenile	drunken	delightful
iterate	repeat	unsettled	impoverish	announce

Pregunta	Opción Correcta	Opción 2	Opción 3	Opción 4
bulwark	rampart	conundrum	festival	confuse
culmination	realization	disaster	serendipity	persuasion
feign	pretend	jab	swoon	dread
auspicious	favorable	deceitful	foreboding	dangerous
gambit	ploy	frolic	testimony	sentence
voracious	ravenous	violent	voluble	rambunctious
facile	glib	ability	section	vindictive
eschew	abstain	revert	accompany	admire
abscond	flee	rob	obscure	absolve
knack	ability	bruise	keepsake	scoundrel
apropos	opportune	unexpected	misspoken	idea
glare	scowl	hide	display	summon
erratic	irregular	enticing	frequent	difficult
civil	public	unkind	trite	questionable
peer	fellow	apple	connote	recluse
fiasco	disaster	festival	happenstance	ceremony
chasm	gorge	charm	bridle	criticize
expertise	mastery	activity	courage	effort
outlandish	absurd	distant	pastoral	belligerent
pine	crave	clean	hate	resolve
exploit	feat	answer	accident	persuade
recalcitrant	unruly	hesitant	subdued	subtract
pretty	terrible	plain	confusing	ugly
coterie	club	various	flirtation	socialize
nefarious	evil	infamous	macabre	distinguished
curry	flatter	spicy	squander	game
preternatural	extraordinary	immature	removed	unearned
pernicious	noxious	illicit	open	undecided
reprisal	retaliation	accusation	loathe	insinuation
manifold	diverse	evident	contemporary	willing
factious	seditious	sham	unreliable	argumentative
drudgery	labor	silliness	evil	investigation
prerequisite	necessary	course	difficult	tar
dire	urgent	questionable	forthright	traitor
grapple	struggle	trap	laugh	intend
sundry	various	aged	supply	tremendous
supplant	replace	grow	undo	question
venerate	revere	ordain	breathe	polish
conciliate	appease	disagree	revive	separate
exultant	jubilant	afraid	expectant	demanding
surreptitious	clandestine	overbearing	indirect	impious
spite	malice	joy	beverage	wonder
scale	climb	sail	swim	skate
smudge	blur	gloat	residue	celebrate
drizzle	sprinkle	curly	sear	drench
mundane	commonplace	dirty	confused	extraordinary
pretension	ambition	stress	waste	strife
affect	cultivate	outcome	share	pompous
herald	hail	insignia	postpone	regal
faculty	gift	defective	school	desire
mirth	glee	anger	sarcasm	mistrust
misprize	despise	devalue	erroneous	covet
impugn	assail	imply	fret	recalcitrant
supervene	follow	intervene	overreach	displace
exigent	urgent	treatise	miser	expedient
fervid	Ardent	delightful	difficult	obstinate
ersatz	artificial	chaotic	impromptu	vague
redolent	odorous	ubiquitous	shy	bellicose
vociferous	strident	numerous	bountiful	garrulous
turpitude	depravity	lethargy	honor	belligerence
homogeneous	alike	strange	polite	alkaline
hub	center	counsel	elder	extension
tame	submissive	lost	evasive	pushy
irk	irritate	shrug	devour	avoid
loom	surface	disappear	cut	teach
fitful	erratic	angry	tired	pronounced
gaudy	flashy	massive	mindful	tasteful
flaunt	parade	conceal	trust	fray
flex	bend	binge	rid	consume
tantalize	flirt	pronounce	reign	equal
dastardly	cowardly	devastating	clever	munificent
aficionado	trickster	novice	devotee	agent
contiguous	adjoining	catching	divided	circumstantial
swindler	charlatan	expert	divinity	debonair
rogue	knave	wander	buffoon	color
apologist	defender	liar	failure	admirer
proxy	delegate	spasm	closeness	court
buffet	strike	protect	barricade	armoire
travesty	mockery	confusion	disaster	speculation
bristle	seethe	aloof	wave	doubt
admonish	caution	laud	decorate	admire
wheedle	plead	retreat	deceive	question
aplomb	poise	mine	clumsiness	complication
aver	state	dissipate	create	hate
mien	carriage	average	vicious	disguise
paroxysm	fit	conundrum	contraction	spite
aegis	sponsorship	superstition	reference	archive
sepulture	burial	parasite	verse	sermon
harridan	witch	governor	vessel	lawyer
apothegm	adage	medicine	speculation	resistance
grandiloquence	bluster	respect	denial	solemnity

Pregunta	Opción Correcta	Opción 2	Opción 3	Opción 4
fulmination	explosion	recession	achievement	blessing
pococurante	blasé	native	hot	hidden
escarpment	cliff	warning	campsite	tomb
plutocrat	banker	priest	judge	astronomer

Cuadro 6: Examen de evaluación 2

.2.3 Examen 3

Pregunta	Opción Correcta	Opción 2	Opción 3	Opción 4
appreciate	respect	agree	contradict	understand
advocate	recommend	insist	appreciate	deplore
qualifications	credentials	measurements	forms	attire
retract	withdraw	deny	apologize	correct
reveal	divulge	explain	discover	recognize
predict	forecast	sure	afraid	explain
quantities	amounts	items	varieties	plates
reduced	decreased	implemented	augmented	reevaluated
constructed	built	guaranteed	located	insured
approximately	nearly	essentially	confidentially	truthfully
recreation	amusement	necessities	education	incidentals
recently	lately	ayuwhere	inside	outside
prepared	ready	anxious	going	afraid
contemporary	concurrent	ancient	modern	several
symbolizes	represents	summarizes	reveals	contains
associates	colleagues	neighbors	superiors	students
discovery	find	election	district	dance
negotiated	arranged	rejectrd	suggested	demanded
conclusive	final	understood	valid	predicted
census	survey	group	number	newspaper
pedestrian	common	unusual	newsworthy	public
fluctuated	varied	rose	stabilized	decreased
controversial	disputatious	popular	personal	biased
compassion	feeling	distrust	revulsion	knowledge
depletion	exhaustion	craving	hoarding	maintenance
aggressive	assertive	depressed	unstable	insecure
dedicated	devoted	polite	agreeable	considerable
demonstrate	show	guess	unnecessary	describe
inappropriate	unsuitable	unnecessary	inconsistent	inarticulate
convincing	persuasive	realistic	reliable	clear
sufficient	adequate	proper	legal	positive
meek	humble	lonely	lazy	loud
fake	simulated	expensive	stolen	sold
potencial	possible	certain	actual	definite
drifted	floated	headed	hurried	returned
primarily	mainly	objectively	actively	subjectively
conflict	disagreement	decision	condition	action
pause	interval	introduction	prayer	play
brisk	lively	short	long	solemn
clumsy	awkward	superior	ignored	noticed
deleted	erased	corrected	circled	determined
ferocious	fierce	hairy	silly	callous
harassing	bothering	asking	bribing	coaxing
garments	clothes	homes	pictures	tools
incompatible	antagonistic	dissimilar	different	incomparable
douse	extinguish	create	prolong	deter
suspect	accussed	nonbeliever	judge	robber
quaint	curious	ancient	elaborate	ultramodern
massive	huge	light	repetitive	infrequent
provisions	clauses	writers	readers	lawyers
relinquish	abandon	hold	control	gain
dissonance	melody	clarity	iscord	volume
issue	subject	book	article	equation
satisfactory	adequate	whimsical	audacious	comprehensive
donate	give	supply	offer	show
mature	grown	intelligent	serious	childlike
rehearse	practice	sing	rewrite	introduce
resumed	continued	returned	repeated	receded
prejudice	bias	anger	rudeness	action
assurance	guarantee	reliance	approval	presence
extinguish	kill	remember	plugin	start
extended	offered	mailed	passed	announced
practically	almost	essentially	always	conveniently
inept	inappropriate	efficient	clever	inferior
alluded	referred	misrepresented	forgot	recounted
zealous	ardent	colorful	rude	clever
enmity	hatred	relationship	friendship	closeness
verbose	redundant	interesting	concise	clever
intransigent	stubborn	honest	friendly	loud
lethargic	indifferent	thirsty	warm	careless
savage	ferocious	friendly	furry	independent
innate	natural	hidden	benevolent	unselfish
distinction	difference	attraction	danger	comfort
fundamental	basic	permanent	interesting	ancient
indicates	suggests	admits	insists	predicates
brilliant	intelligent	known	professional	popular
exactly	precisely	intuitively	immediately	briefly

Pregunta	Opción Correcta	Opción 2	Opción 3	Opción 4
discourteous	rude	handsome	irritable	lost
excludes	ban	avoid	laugh	invite
gist	point	ending	length	title
disappear	vanish	behave	forget	happy
synthetic	artificial	expensive	fashionable	new
hazardous	dangerous	challenging	slippery	exciting
tranquil	peaceful	queasy	sad	sleepy
introverted	reserved	forward	obnoxious	cautions
assassinated	murdered	honored	elected	impeached
expansion	enlargement	problems	painting	decrease
charged	ran	fell	came	loped
audible	heard	seen	felt	ignored
essential	integral	possible	minor	negative
horrendous	dreadful	delightful	spectacular	obscene
blatantly	openly	hardly	offensively	extremely
furtive	secretive	unusual	sleepy	sickly
kindred	related	kindly	polite	humorous
ludicrous	comical	loud	somber	common
nuances	subtleties	images	rhythm	rhymes
opportune	appropriate	awaited	lucky	anticipated
tantalize	tempt	dissuade	inform	fool
obliterated	erased	covered	produced	outlined
sealed	closed	hidden	unreachable	ancient
reproduced	printed	taken	renewed	restored
inaccessible	impenetrable	desolate	high	dry
imitate	copy	criticize	admire	remember
granted	assumed	hoped	guaranteed	convinced
captivated	enchanted	frightened	bored	disgusted
precede	before	point	indicate	cause
severity	seriousness	purpose	location	perpetrator
collect	accumulate	bury	desire	require
reliable	dependable	automatic	versatile	fast
typical	characteristic	complete	total	hard
unaccustomed	unused	surprised	disappointed	afraid
ashamed	humiliated	unhappy	disappointed	disgusted

Cuadro 7: Examen de evaluación 3