Benemérita Universidad Autónoma de Puebla

Facultad de Ciencias de la Computación

Desarrollo de un Modelo para encontrar la Similitud Semántica Multilingüe

Tesis Profesional para obtener el Título de:
Licenciatura en Ingeniería en Ciencias de la Computación

Presenta:
Emanuel Aguilar Benítez

Asesora:
Dra. Darnes Vilariño Ayala

Coasesora:
Dra. Mireya Tovar Vidal

Primavera 2015

El presente trabajo de tesis lo dedico a mi familia, quienes siempre y sin importar que, me han apoyado continuamente a lo largo de mi vida y mi carrera. Gracias Papá, Mamá y Makito. Siempre les estaré agradecido por todo.

Agradezco a mis asesoras Darnes Vilariño Ayala y Mireya Tovar Vidal por todo el apoyo, conocimiento, tiempo y paciencia que me han regalado. También doy las gracias por la ayuda que me brindo la Vicerrectoría de Investigación y Estudios de Posgrado.

Por ultimo agradezco a mi novia y amigos que me han ayudado a seguir siempre adelante.

Resumen

La similitud semántica ha sido objeto de estudio durante muchos años dentro del área del Procesamiento del Lenguaje Natural, esto debido a su amplio rango de aplicaciones, como por ejemplo: máquinas de traducción, construcción automática de resúmenes, atribución de autoría, pruebas de lectura comprensivas, recuperación de información y muchas otras que necesitan medir el grado de similitud entre dos textos dados.

El objetivo principal de este trabajo de tesis consiste en desarrollar un modelo capaz de detectar el grado de similitud semántica entre un par de sentencias. Para dar solución a este problema se proponen dos modelos, el primero basado en aprendizaje supervisado y un segundo modelo basado en aprendizaje no supervisado. Estos modelos se implementaron mediante el lenguaje Python, además se utilizaron las herramientas Clips Pattern, NetworkX, WordNet y OpenThesaurus-es.

El primer modelo utiliza 16 características para entrenar un clasificador, ya sea máquina de soporte vectorial, Naïve Bayes o perceptrón simple, este modelo obtuvo como mejores resultados un 35.16% de precisión para el idioma inglés y 64.61% para el idioma español. El segundo modelo se basa en la reconstrucción de una de las sentencias por medio de la otra empleando los sinónimos de las palabras que las componen, con este modelo se obtuvo una precisión del 37.86% para el idioma inglés y 70.76% para el idioma español.

Índice General

Índice de Figuras

Índice de Tablas

Capítulo 1 Introducción

La similitud semántica textual tiene como objetivo determinar qué tan semejantes son los sentidos de dos textos. Este concepto difiere de encontrar el grado de similitud textual, que está solamente interesado en medir el número de componentes léxicas que comparten ambos textos, es decir, el valor de similitud textual puede variar cuando uno de los textos no está completo y no se logra con la poca información medir cuanto realmente comparten.

Encontrar la similitud semántica entre pares de textos se ha convertido en un gran reto para los especialistas en Procesamiento de Lenguaje Natural (PLN), ya que se puede aplicar en diferentes tareas de PLN tales como máquinas de traducción, construcción automática de resúmenes, atribución de autoría, pruebas de lectura comprensivas, recuperación de información y muchas otras, que necesitan medir el grado de similitud entre dos textos dados.

Durante el primer capítulo de esta tesis se presentan algunos trabajos relacionados a esta tarea, el planteamiento del problema y los objetivos a cumplir. En el segundo capítulo se describen los conceptos y herramientas que se emplearon para el desarrollo de ambos modelos. El objetivo del tercer capítulo es explicar a detalle el funcionamiento de cada modelo propuesto. Los resultados que se obtuvieron al aplicar ambos modelos a un conjunto de datos de prueba son revelados en el capítulo 4. Por último se dan las conclusiones sobre la realización de esta tesis.

1.1 Antecedentes

Se han desarrollado diversas investigaciones para detectar el grado de similitud entre pares de textos, a continuación se presentan las más relevantes asociadas a los trabajos presentados en el marco de la Conferencia SemEval 2013[1].

En el trabajo desarrollado en [1], se describe un sistema en el cual para estimar la similitud semántica entre dos enunciados se usan modelos de regresión que incluyen las siguientes características:

- *n-gramas* repetidos entre enunciados
- similitud léxico-semántica entre palabras distintas
- métricas de similitud de cadenas
- similitud de contenido afectivo
- longitud del enunciado

Todas las características mencionadas anteriormente se combinan usando uno de los dos modelos siguientes:

- El primer modelo es un modelo de regresión múltiple linear:

$$\hat{D}_L = a_0 + \sum_{n=1}^{k} a_n \, f_k,$$

Donde $\hat{D}_L$ es la similitud estimada, f_k son las métricas de similitud semántica sin supervisión y a_n son los parámetros entrenables del modelo.

[1] http://www.cs.york.ac.uk/semeval-2013/index.php?id=tasks

- El segundo modelo detecta la relación que en cuanto a longitud poseen cada una de las sentencias.

Esta propuesta obtiene un 47% de aciertos reportados con los datos del SemEval 2013.

En el trabajo presentado en [2], se emplea un modelo lineal estimado usando regresión *ridge*. Se analiza el comportamiento de la propuesta aplicando validación cruzada de 5 pliegues con los datos de entrenamiento. Esto permite afinar la sanción α que se emplea en la regresión *ridge*, con $\alpha \in 2^{\{-5,-4,\ldots,4\}}$. Dadas dos sentencias de entrada, S_1 y S_2, el sistema extrae:

- Características de superposición de n-gramas
- Características de longitud
- Características de sentimientos

Posteriormente el sistema estandariza los valores de las características, substrayendo la media de las características y dividiendo por su desviación estándar. La media y desviación estándar son estimadas del conjunto de entrenamiento, o de cada partición de entrenamiento durante la validación cruzada. De igual manera se usan adaptaciones al dominio para facilitar la generalización a nuevos dominios. En lugar de tener un sólo peso para cada una de las características descritas anteriormente, el sistema cuenta con valores genéricos que se activan en el momento en que un nuevo conjunto, distinto a los usados en la fase de entrenamiento, aparece. Este sistema presenta en promedio una tasa de aciertos del 45.03%.

El sistema que se describe en [3] desambigua el sentido de las palabras usando contextos, en este, las palabras en los enunciados son asignadas con el sentido apropiado utilizando sus contextos. La similitud de enunciados es calculada con el número de sentidos que comparten, es razonable asumir que enunciados similares deben tener más sentidos superpuestos.

Para determinar la superposición del sentido se comparan las características de las palabras, primero se conceptualizan las palabras y después se calcula su similitud, basándose en una estructura jerárquica que se basa en WordNet; cada palabra en un enunciado es asignada a un sentido en el diccionario WordNet. Para reducir las limitaciones del diccionario, se utilizan los términos de los sentidos relacionados con la palabra objetivo, se asume que las palabras que co-ocurren en un enunciado comparten relación del sentido, y mientras más similares sean los enunciados compartirán más términos las definiciones de sus palabras. Por ello no sólo se extraen los términos del enunciado principal, sino que también, para cada palabra del enunciado principal, se extraen los términos de su *hypernym, hyponym, meronym, holonym y troponym*; con esto se forma un conjunto "contexto".

Finalmente se compara el contexto del enunciado con diferentes conjuntos de contextos para determinar cuál sentido debe ser asignado a las palabras. La implementación de este sistema reporta un 41.51% de aciertos.

El sistema "*Semantic Text Similarity by use of Knowledge Bases*" propuesto en [4] procesa características que se obtienen de diferentes bases de conocimiento como son WordNet, Wikipedia y Wiktionary. Las puntuaciones de similitud derivadas de estas características son introducidas dentro de varios perceptrones multicapa. Dependiendo del tamaño de los textos a comparar se usan diferentes parámetros para las redes neuronales; esto es que para cada grupo de longitud de enunciados, los pesos del perceptrón multicapa se calcularon separadamente.

Para modelar cada red neuronal usaron la biblioteca de código abierto "*Neuroph*", cada perceptrón fue definido con 48 capas de entrada, correspondientes a las puntuaciones extraídas de las características, 4 capas ocultas y una capa de salida que representa la puntuación de similitud entre los enunciados. En la fase de entrenamiento la salida final de las redes neuronales es comparada con datos de jueces humanos. Además de las características que se obtienen de WordNet, Wikipedia y Wiktionary aplicaron características de números y expresiones financieras y características de *n-gramas*. El sistema se desempeña bastante bien cuando los textos de entrada son cortos y de tamaños semejantes, pero cuando existe una variación considerable de tamaño entre estos, o son textos muy largos el desempeño del sistema cae considerablemente. La media de aciertos en este sistema es de 0.16.

Por último, en el sistema "*Textual Similarity based on Lexical-Semantic Features*" que se presenta en [5] se usan diferentes tipos de características léxicas y semánticas para entrenar un clasificador de bolsas que se utiliza para decidir la similitud entre los enunciados. Se implementaron tres variaciones del sistema cada una de estas variaciones utiliza un grupo particular de características. Cada par de enunciados es tokenizado, lematizado y post-etiquetado usando la herramienta "Freeling 2.2". Posteriormente varios métodos y algoritmos son aplicados para extraer todas las características necesarias para el sistema de máquina de aprendizaje. La primer variación llamada MultiSemLex, toma en cuenta todas las características extraídas y entrena un modelo con un clasificador de bolsas; la segunda variación llamada MultiLex y la tercera llamada MultiSem usan el mismo clasificador, pero incluyen diferentes características. MultiLex utiliza características extraídas de métricas léxico-semánticas y alineamiento léxico-semántico. Por otra parte, MultiSem, utiliza características extraídas solamente de alineación semántica. El sistema obtuvo un coeficiente de correlación general de 0.61.

A pesar de que se ha avanzado en la detección del grado de similitud semántica entre pares de sentencias, no se han logrado resultados muy significativos para el idioma inglés, y peor aún para el idioma español, para este último se dispone de un corpus de entrenamiento de sólo 65 pares.

1.2 Planteamiento del Problema

El objetivo es dado un par de sentencias S_1 y S_2, ofrecer un valor entre 0 y 5 que mida el grado de similitud entre ambas sentencias. Se pretende trabajar con sentencias en los idiomas inglés y español. Los valores que representan el grado de similitud son explicados en la Tabla 1.1 que se muestra a continuación.

Tabla 1.1. Valores representativos del grado de similitud

Representación del Grado de Similitud		
Valor	**Significado**	**Ejemplo**
5	Las dos sentencias son completamente equivalentes, ya que significan lo mismo.	*the bird is bathing in the sink* *birdie is washing itself in the water basin*
4	Las dos sentencias son en su mayoría equivalentes, pero difieren algunos detalles sin importancia.	*in May 2010, the troops attempted to invade Kabul* *the US army invaded Kabul on May 7th last year, 2010*
3	Las dos sentencias son más o menos equivalentes, pero alguna información importante difiere o está ausente en una sentencia.	*John said he is considered a witness but not a suspect* *"he is not a suspect anymore" John said*
2	Las dos sentencias no son equivalentes, pero comparten algunos detalles.	*they flew out of the nest in groups* *they flew into the nest together*
1	Las dos sentencias no son equivalentes, pero tratan sobre el mismo tema.	*the woman is playing the violin* *the young lady enjoys listening to the guitar*
0	Las dos sentencias tratan sobre temas diferentes.	*John went horseback riding at dawn with a whole group of friends* *sunrise at dawn is a magnificent view to take in if you wake up early enough for it*

Encontrar las características adecuadas que nos permitan representar a ambas sentencias es uno de los objetivos fundamentales del presente trabajo de tesis. Se han utilizado representaciones basadas en bolsas de palabras que comparten, *n-gramas* tanto de caracteres como de palabras que comparten, sin embargo los resultados reportados a nivel internacional no superan el 83 %.

En la presente tesis se pretende probar diferentes tipos de características buscando con ello detectar el sentido de cada oración, por lo que nos planteamos las siguientes preguntas de investigación:

¿Qué características nos permiten medir de mejor manera cuando dos textos muestran una total equivalencia, o ninguna equivalencia?
¿Las características analizadas se comportan de manera diferente de acuerdo al idioma que se está analizando?
¿La representación de las sentencias mediante grafos, permiten recuperar el sentido de las sentencias?

Para dar respuesta a las preguntas de investigación, nos hemos planteado los siguientes objetivos:

1.3 Objetivos

Objetivo general

Desarrollar un modelo para detectar el grado de similitud semántica multilingüe.

Objetivos particulares

1. Analizar las diferentes propuestas enviadas al SemEval 2013.
2. Analizar diferentes características para la representación de los pares de sentencias para el idioma inglés y para el idioma español.

3. Desarrollar un modelo que permita obtener el grado de similitud semántica entre un par de sentencias, utilizando las características seleccionadas sobre los datos de entrenamiento proporcionados en el SemEval 2014, tanto para el idioma inglés como para el idioma español.
4. Clasificar los datos de prueba con el modelo desarrollado.

1.4 Justificación

Como se ha comentado con anterioridad el desarrollo de herramientas automáticas para el procesamiento del Lenguaje Natural ha tomado mucho auge en los últimos años, lograr detectar si dos textos expresan lo mismo es importante para tereas tales como: Recuperación de Información, sistemas de lecturas comprensivas, atribución de autoría entre otras. A pesar de que se han realizado grandes esfuerzos, los resultados más importantes se han obtenido para el idioma inglés, no así para el idioma español, es por ello que en la presente tesis se pretende estudiar diferentes características representativas y evaluar su comportamiento en los dos idiomas, utilizando el mismo modelo.

1.5 Organización de la Tesis

El presente trabajo de tesis está formado por cinco capítulos, que se enuncian a continuación:

- **Capítulo 1 Introducción:** En este capítulo se mencionan algunos trabajos relacionados a la similitud semántica en el marco de la conferencia SemEval 2013, además de detallar el problema a resolver y los objetivos a cumplir.

- **Capítulo 2 Marco Teórico:** El objetivo de este capítulo es presentar los conceptos y herramientas empleados durante la realización de este trabajo de tesis.

- **Capítulo 3 Modelos:** Aquí se describen a detalle los modelos propuestos para determinar la similitud semántica entre dos sentencias, se propone un conjunto de características con las que se entrenará un modelo basado en aprendizaje supervisado y se explica un modelo que reconstruye las oraciones basándose en las palabras y sinónimos que comparten.

- **Capítulo 4 Análisis de Resultados:** Se exponen los resultados que se obtuvieron en la implementación de ambos modelos, tanto para el idioma español como el inglés.

- **Conclusiones y Recomendaciones:** Se dan las conclusiones conforme a las pruebas realizadas, además de las recomendaciones a considerar para trabajos futuros.

Capítulo 2 Marco Teórico

En este capítulo se describen de manera general las herramientas y conceptos estudiados para la realización del presente trabajo de tesis.

2.1 Minería de Datos

El término minería de datos es aplicado al conjunto de técnicas y tecnologías que permiten explorar grandes bases de datos, de manera automática o semiautomática, con el objetivo de encontrar patrones repetitivos, tendencias o reglas que expliquen el comportamiento de los datos en un determinado contexto [6].

Básicamente, la minería de datos surge para intentar ayudar a comprender el contenido de un repositorio de datos. De forma general, los datos son la materia prima bruta. En el momento que el usuario les atribuye algún significado especial pasan a convertirse en información. Cuando los especialistas elaboran o encuentran un modelo, haciendo que la interpretación que surge entre la información y ese modelo represente un valor agregado, entonces nos referimos al conocimiento.

2.2 Minería de Textos

La minería de textos es una disciplina especializada en la obtención de información que se encuentra de forma implícita en un conjunto de textos. Esto es posible a través de la identificación de patrones y correlaciones de los términos contenidos en ellos [7]. Existen varias aplicaciones para la minería de textos, de las cuales destacan:

- Extracción de información
- Análisis de sentimientos o minería de opiniones
- Clasificación de documentos
- Elaboración de resúmenes
- Extracción de conocimiento

2.3 Procesamiento de Lenguaje Natural

También conocido como lingüística computacional, es un campo multidisciplinario de la lingüística y la computación que utiliza la última para estudiar y tratar el lenguaje humano. Para lograrlo, intenta modelar de forma lógica el lenguaje natural desde un punto de vista computacional. Dicho modelado no se centra en ninguna de las áreas de la lingüística en particular, sino que es un campo interdisciplinar, en el que participan lingüistas, especialistas en inteligencia artificial, psicólogos cognoscitivos y expertos en lógica, entre otros [8].

Tradicionalmente el lenguaje natural se divide en seis niveles: fonética/fonología, morfología, sintaxis, semántica, pragmática y discurso. No existen criterios exactos para la separación de cada uno de los niveles; de hecho, las diferencias entre los niveles se basan en el enfoque de análisis de cada uno [9]. A continuación se describe brevemente cada nivel del lenguaje.

2.3.1 Fonética / Fonología

La fonética es la parte de la lingüística que se dedica a la exploración de las características del sonido, el cual es un elemento substancial del lenguaje. Eso determina que los métodos de fonética sean en su mayoría físicos; por eso su posición en la lingüística es bastante independiente. Los problemas en fonética computacional están relacionados con el desarrollo de sistemas de reconocimiento de voz y síntesis del habla.

2.3.2 Morfología

El área de la morfología se enfoca en el estudio de la estructura interna de las palabras (sufijos, prefijos, raíces, flexiones) y el sistema de categorías gramaticales de los idiomas (género, número, etc.). Los problemas de morfología computacional están relacionados con el desarrollo de sistemas de análisis y síntesis morfológica automática.

2.3.3 Sintaxis

La sintaxis se dedica a analizar las relaciones entre las palabras dentro de la frase. La sintaxis computacional debe tener métodos para análisis y síntesis automática, es decir, construir la estructura de la frase, o generar la frase basándose en su estructura.

2.3.4 Semántica

El propósito de la semántica es "entender" la frase, esto es, conocer el sentido de las palabras e interpretar las relaciones sintácticas. Otra tarea de la semántica es definir los sentidos de las palabras. Una aplicación importante del análisis semántico es la *desambiguación automática de sentidos de las palabras*. Por ejemplo *gato* puede hacer referencia a *un animal*, *una herramienta* o *una persona*. Para saber cuál de los sentidos se usa en un contexto dado se pueden aplicar diferentes métodos con el fin de analizar las demás palabras presentes en el contexto.

2.3.5 Pragmática

Usualmente se dice que la pragmática trata de las relaciones entre la oración y el mundo externo, se puede decir que lo que interesa a la pragmática son las intenciones del autor del texto o del hablante. Por ejemplo, en la frase *¿me puedes pasar la sal?*, se pregunta sobre la posibilidad de pasarla, siendo la intención de pedir la sal. Otro ejemplo del dominio de la pragmática es la clase de oraciones que tienen como característica particular ser acciones por sí mismas. Por ejemplo, decir *prometo* es precisamente la acción de *prometer*.

2.3.6 Discurso

El discurso se forma por varias oraciones hiladas, esas oraciones tienen ciertas relaciones entre sí. La tarea principal del análisis del discurso consiste en la resolución de la correferencia o relaciones anafóricas. Por ejemplo, en el discurso *"he visto una nueva casa ayer. Su cocina era excepcionalmente grande"* (su = de la casa); o *"llegó Juan. Él estaba cansado"* (él = Juan). Esas son relaciones anafóricas, y la computadora tiene que interpretarlas correctamente para poder construir las representaciones semánticas.

2.4 Similitud Semántica

La similitud semántica ha sido objeto de estudio, durante muchos años, dentro del área de la recuperación de información. El cálculo de la similitud semántica es un procedimiento genérico en una gran variedad de aplicaciones en áreas de computación, lingüística e inteligencia artificial [10]. Ejemplo de ello podría ser su uso en tareas de procesamiento de lenguaje natural, desambiguación de palabras, detección y corrección de errores en la escritura (malapropismo), clasificación de textos, etc.

2.5 Medidas de Similitud

En procesamiento de lenguaje natural y minería de textos se suelen utilizar métricas que evalúan las características de cada texto para así obtener un índice de similitud; estas métricas por lo general devuelven un valor entre 0 y 1, donde 0 significa que son totalmente diferentes y 1 que son iguales. A continuación se mencionan algunas de las métricas de similitud que tienen un mejor desempeño.

2.5.1 Similitud Coseno

La similitud coseno [11], es una medida de la similitud existente entre dos vectores *a* y *b* de dimensión *N* en un espacio que posee un producto interior con el que se evalúa el valor del coseno del ángulo comprendido entre ellos. De acuerdo con la ecuación (1), esta función trigonométrica proporciona un valor igual a 1 si el ángulo comprendido es 0, es decir si ambos vectores apuntan a un mismo lugar. Con cualquier ángulo existente entre los vectores, el coseno arrojaría un valor inferior a 1. Si los vectores fueran ortogonales el coseno se anularía, y si apuntasen en sentido contrario su valor sería -1. De esta forma, el valor de esta métrica se encuentra entre -1 y 1; aunque en la mayoría de los casos, por razones prácticas, este valor se normaliza para que sus valores se encuentren en el rango de [0,1].

$$similitud\ coseno(a,b) = \frac{\sum_{i,j}^{N} a_i b_j}{\sqrt{\sum_{i}^{N} a_i^2}\sqrt{\sum_{j}^{N} b_j^2}} \qquad (1)$$

Donde, *a* y *b* son vectores de dimensión *N*.

Esta medida se emplea frecuentemente representando las palabras en un espacio vectorial. En minería de textos se aplica la similitud coseno con el objetivo de establecer una métrica de semejanza entre textos.

2.5.2 Similitud Euclidiana

Esta similitud se obtiene restándole a 1 la distancia euclidiana existente entre los vectores *p* y *q* de dimensión *N*. La distancia euclidiana está basada en el teorema de Pitágoras; como se puede observar en la ecuación (2), dicha distancia es la raíz cuadrada de la suma de los cuadrados de cada una de las diferencias, cada una puede ser considerada un cateto, para así obtener la distancia (hipotenusa) entre esos dos puntos [12].

$$distancia\ euclidiana(p,q) = \sqrt{\sum_i^N (p_i - q_i)^2} \qquad (2)$$

Donde, p y q son vectores de dimensión N.

Empleando la ecuación (2), se puede representar la similitud euclidiana como se muestra a continuación en la ecuación (3).

$$similitud\ euclidiana(p,q) = 1 - \sqrt{\sum_i^N (p_i - q_i)^2} \qquad (3)$$

Donde, p y q son vectores de dimensión N.

2.5.3 Similitud Levenshtein

La similitud Levenshtein, similar a la anterior, se calcula restándole a 1 la distancia Levenshtein.

La distancia de Levenshtein [13], distancia de edición o distancia entre palabras (creada por Vladimir Levenshtein en 1965), es el número mínimo de operaciones requeridas para transformar una cadena de caracteres en otra. Se entiende por operación a, una inserción, eliminación o sustitución de un carácter.

Cuanto más corta sea la distancia entre dos cadenas, más parecidas son estas. Si la distancia es 0, las dos cadenas son iguales.

El algoritmo para hallar la distancia de Levenshtein, se basa en crear una matriz de enteros que tenga tantas filas como la longitud de una de las cadenas más uno, y tantas columnas como la longitud de la otra cadena más uno. La primer fila y la primer columna se rellenan con los valores 0, 1, 2, 3,.., hasta llenarlas por completo, y a partir de ahí se va rellenando el resto de la matriz sumando costes de transformaciones y quedándose siempre con el mínimo. Como la matriz está completamente llena, la última casilla de la matriz dará el coste de la transformación.

2.5.4 Similitud Dice

Esta similitud está basada en un estadístico utilizado para comparar la similitud de dos muestras. La fórmula original de Sorensen [14] estaba destinada a ser aplicada a la presencia/ausencia de datos y está definida de la siguiente forma:

$$QS = \frac{2C}{A+B} = \frac{2|A \cap B|}{|A|+|B|} \qquad (4)$$

Donde en la ecuación (4), A y B son el número de especies en las muestras A y B, respectivamente, y C es el número de especies compartidas por las dos muestras; QS es el cociente de similitud y varía de 0 a 1. Esta expresión se extiende fácilmente a la abundancia en lugar de la presencia/ausencia de especies. El índice de Sorensen es idéntico al coeficiente de Dice que siempre está en el rango de 0 a 1.

En minería de textos y procesamiento de lenguaje natural, se emplean como muestras conjuntos de *n-gramas*, para así determinar la cantidad de *n-gramas* que comparten los textos de entrada y con esto proporcionar un grado de similitud.

2.5.5 Intertextualidad

La intertextualidad entre dos textos *a* y *b*, se define como el porcentaje de *n-gramas* distintos contenidos en *a* y que también se encuentran en *b*. La intertextualidad puede ser usada como medida de similitud, ya que mientras mayor cantidad de *n-gramas* se comparta entre los textos, mayor será su equivalencia.

2.6 Grafos

Un grafo en el ámbito de las ciencias de la computación es una estructura de datos, en concreto un tipo abstracto de datos, que consiste en un conjunto de nodos, también llamados vértices, y un conjunto de arcos o aristas que establecen relaciones entre los nodos. El concepto de grafo como tipo abstracto de datos desciende directamente del concepto matemático de grafo.

Informalmente se define como $G = (V, E)$, siendo los elementos de V los vértices, y los elementos de E, las aristas (*edges* en inglés). Formalmente, un grafo, G, se define como un par ordenado, $G = (V, E)$, donde V es un conjunto finito y E es un conjunto que consta de dos elementos de V [15].

2.6.1 Isomorfismo de grafos

En teoría de grafos, se dice que dos grafos G_1 y G_2 son isomorfos cuando teniendo en cuenta la apariencia diferente realmente son iguales, porque coinciden en:

- El número de lados
- El número de vértices
- El conjunto de valencias
- Ser o no conexos
- El número de circuitos de longitud *n*
- Tener o no circuito de Euler.

Esto implica que todos los vértices de G_1 tienen un vértice equivalente en G_2, y que todas las aristas del grafo G_1 tienen una arista equivalente en G_2. La consecuencia de esto es que con las propiedades de un vértice en G_1 como argumentos, y por medio de una función biyectiva de *f*, se puede obtener un vértice en G_2 con las mismas propiedades. También teniendo las propiedades de un vértice en G_2 como argumentos y por medio de una función biyectiva *g*, se puede obtener un vértice en G_1 con las mismas propiedades.

Por otro lado, se sabe que dos grafos G_1 y G_2 son isomorfos si y solo si para alguna ordenación de vértices y aristas, sus matrices de incidencia son iguales [16].

2.6.2 Centralidad

Se refiere a la posición de los nodos en el grafo, y la centralización al conjunto de la estructura del grafo [17]. La centralidad mide la contribución de un nodo según su ubicación en la red, independientemente de si se esté evaluando su importancia, influencia, relevancia o prominencia. Existen distintos algoritmos de centralidad [18], pero en particular cuando se busca conocer los nodos que poseen la mayor cantidad de enlaces es

recomendable aplicar la centralidad de grado (*degree centrality*), ya que esta centralidad corresponde al número de enlaces que posee un nodo con respecto a los demás.

2.7 Clasificadores

En el campo del Procesamiento de Lenguaje Natural, la tarea de clasificación consiste en asignar un texto a una categoría específica de entre un conjunto de categorías. Los algoritmos de clasificación "aprenden" con ejemplos etiquetados en forma manual o por medio de algún proceso automático. Los algoritmos de clasificación construyen un modelo de términos individuales y otras características como longitud o estructura. Al final el modelo puede ser usado para categorizar nuevos documentos [19]. A continuación se describen algunos clasificadores.

2.7.1 Máquina de Soporte Vectorial

Una Máquina de Soporte Vectorial (o SVM por sus siglas en inglés *Support Vector Machine*) es un conjunto de algoritmos de aprendizaje supervisado desarrollado por Vladimir Vapnik. La SVM está basada en la minimización de riesgo estructural. En diversas aplicaciones ha demostrado gran eficiencia comparada con otras máquinas de aprendizaje tradicional como son las redes neuronales y principalmente utilizada para resolver problemas de clasificación.

Su funcionamiento es el siguiente: La SVM primero mapea los puntos de entrada a un espacio de características de una dimensión mayor (por ejemplo, si los puntos de entrada están en R^2 entonces son mapeados por la SVM a R^3) y encuentra un hiperplano que los separe y maximice el margen *m* entre las clases en este espacio como se visualiza en la Figura 2.1.

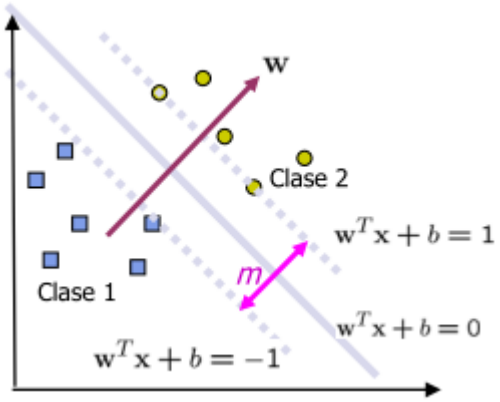

Figura 2.1. La frontera de decisión debe estar lo más lejos posible de los datos de ambas clases.

Sin ningún conocimiento del mapeo, la SVM encuentra el hiperplano óptimo utilizando el producto punto con funciones en el espacio de características que son llamadas *kernels*, las cuales ofrecen la solución a casos con más de dos variables predictoras, curvas no lineales de separación, etc. La solución del hiperplano óptimo puede ser escrita como la combinación de unos pocos puntos de entrada que son llamados vectores de soporte [20].

2.7.1.1 Función Kernel

Se conoce como *kernel*, a las funciones de similitud que habilitan a una máquina de soporte vectorial a operar en un espacio de alta dimensionalidad de características implícitas. Esta función disminuye el costo computacional al no calcular las coordenadas de los datos en el espacio de alta dimensionalidad, esto se logra, simplemente calculando el producto punto entre las imágenes de todos los pares de datos en el espacio de características. Dentro de las distintas funciones *kernel* existentes están el *kernel* lineal, las funciones polinomiales y *"radial basis function"* (RBF), que son las más frecuentemente utilizadas en diversas aplicaciones (Ver figura 2.2) [21].

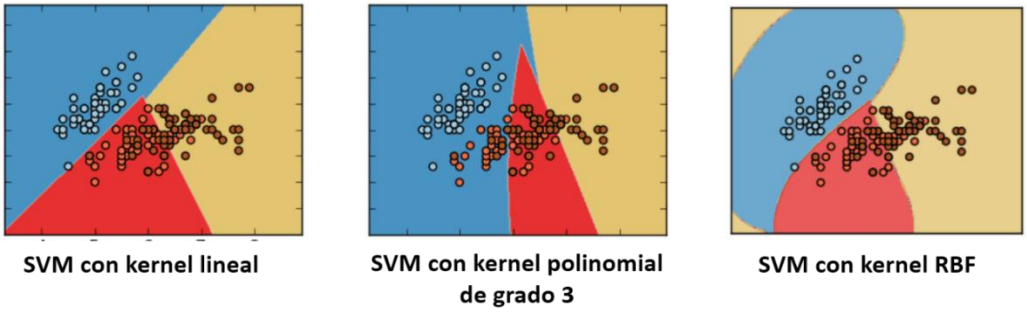

Figura 2.2. Ejemplos de tipos de kernel en una SVM.

Las funciones mencionadas se definen de la siguiente forma:

1. **Kernel lineal:**
   $K(x_i, x_j) = x_i^T \cdot x_j$
2. **Kernel polinomial:**
   $K(x_i, x_j) = (x_i \cdot x_j + 1)^d$, $donde\ d\ \epsilon\ \mathbb{N}$ es el grado del polinomio.
3. **Kernel *Radial basis function*:**
   $K(x_i, x_j) = exp\left(-\frac{\|x_i - x_j\|^2}{2p^2}\right), donde\ p > 0$ es el parámetro que controla el ancho del *kernel*.

2.7.2 Naïve Bayes

Es uno de los clasificadores más utilizados por su simplicidad y rapidez. Se trata de una técnica de clasificación y predicción supervisada que construye modelos que predicen la probabilidad de posibles resultados [22]. Dicha técnica está basada en el Teorema de Bayes, en cuanto al funcionamiento del algoritmo éste se destaca en dos partes:

- Primera fase: construcción del modelo, el cual requiere de cuatro pasos.
  - ✓ Calcular las probabilidades a priori de cada clase.
  - ✓ Para cada clase, realizar un recuento de los valores de atributos que toma cada ejemplo. Se debe distribuir cada clase por separado para mayor comodidad y eficiencia del algoritmo.
  - ✓ Aplicar la corrección de Laplace, para que los valores cero no den problemas.
  - ✓ Normalizar para obtener un rango de valores [0,1].

- Segunda fase: Clasificar un nuevo ejemplo *x*.
  - ✓ Para cada clase disponible, se determinan los valores de probabilidad de cada valor de los atributos del nuevo ejemplo.
  - ✓ Aplicar la fórmula de Naïve Bayes (ver ecuación (5)).

$$P(C_i|X) = \frac{P(X|C_i)\,P(C_i)}{P(X)} \qquad (5)$$

Donde, *X* representa el vector de características de cada ejemplo *x* a clasificar, *C* representa el vector de posibles clases, $P(C_i)$ son las probabilidades a priori de cada clase, $P(X|C_i)$ es la probabilidad de que en la clase C_i se encuentren cada una de las características *X*, $P(C_i|X)$ son las probabilidades a posteriori de la pertenencia de las características *X* a la clase C_i.

2.7.3 Red Neuronal

Las redes neuronales son un campo muy importante dentro de la inteligencia artificial inspirándose en el comportamiento conocido del cerebro humano sobre las neuronas y sus conexiones. Se trata de crear modelos artificiales que solucionen problemas difíciles de resolver mediante técnicas algorítmicas convencionales [23].

Las características fundamentales de las Redes Neuronales Artificiales (RNA) son [24]:

- **Aprenden de la experiencia:** Las RNA pueden modificar su comportamiento como respuesta a su entorno. Dado un conjunto de entradas, las RNA se ajustan para producir respuestas consistentes. Una amplia variedad de algoritmos de entrenamiento se han desarrollado, cada uno con sus propias ventajas e inconvenientes.
- **Generalizan de ejemplos anteriores a los ejemplos nuevos:** Una vez que la RNA esté entrenada, la respuesta de la red puede ser, hasta un cierto punto, insensible a pequeñas variaciones en las entradas, lo que las hace idóneas para el reconocimiento de patrones.
- **Abstracción de la esencia de las entradas:** Algunas RNA son capaces de abstraer información de un conjunto de entradas. Por ejemplo, en el caso de reconocimiento de patrones distorsionados de una letra. Una vez que la red sea correctamente entrenada será capaz de producir un resultado correcto ante una entrada distorsionada, lo que significa que ha sido capaz de aprender algo que nunca había visto.

2.7.3.1 Perceptrón Simple

Desarrollado por Frank Rosenblatt, es un sistema capaz de realizar tareas de clasificación de forma automática [25], es reconocido por su capacidad de aprender a reconocer patrones. Es una red neuronal monocapa con varias neuronas de entrada conectadas a la neurona de salida. El valor de la salida se puede calcular de acuerdo a la ecuación (6), donde *y* tomará un valor de +1 si, la suma de los productos de las entradas multiplicadas por sus pesos más el valor del umbral es mayor a cero; en otro caso *y* toma un valor de -1.

$$y = \begin{cases} +1, \ si & w_1x_1 + w_2x_2 + \theta > 0 \\ -1, \ si & w_1x_1 + w_2x_2 + \theta \leq 0 \end{cases} \qquad (6)$$

El perceptrón simple es un hiperplano de dimensión *n-1* capaz de separar las clases. En el caso de que la salida sea +1, la entrada pertenecerá a una clase, situada a un lado del hiperplano. En el caso de que a salida sea -1, la entrada pertenecerá a la clase contraria, situada al otro lado del hiperplano.

2.8 Herramientas

En esta sección se describen de manera breve las herramientas que fueron utilizadas durante la implementación de los modelos que son expuestos en el siguiente capítulo.

2.8.1 WordNet

WordNet [26] es una gran base de datos léxica para el idioma inglés. Sustantivos, verbos, adjetivos y adverbios están agrupados dentro de conjuntos de sinónimos cognitivos llamados "*synsets*", cada uno expresa un concepto distinto. Los *synsets* están vinculados entre sí por medio de relaciones conceptuales semánticas y léxicas. La red resultante de palabras y conceptos semánticos se puede visualizar con el navegador. WordNet es libre y está disponible públicamente para su descarga. La estructura de WordNet hace que sea una herramienta útil para la lingüística computacional y el procesamiento de lenguaje natural.

Superficialmente WordNet se asemeja a un tesauro[2], en que agrupa palabras basadas en sus significados. Sin embargo, hay algunas diferencias importantes. En primer lugar, WordNet no sólo interconecta formas de palabra sino que especifica sentidos a las palabras. Como resultado, las palabras que se encuentran en estrecha proximidad la una de la otra en la red son semánticamente desambiguas. En segundo lugar, WordNet etiqueta las relaciones semánticas entre palabras, mientras que las agrupaciones de palabras en un tesauro no siguen ningún patrón explícito distinto de la similitud de significado.

2.8.1.1 Estructura

La principal relación entre palabras en WordNet es la sinonimia, como entre las palabras *shut* y *close* o *car* y *automobile*. Los sinónimos son agrupados en conjuntos no ordenados conocidos como *synsets*.

Cada uno de los 117 000 *synsets* de WordNet está vinculado a otros *synsets* por medio de un pequeño número de "relaciones conceptuales". Adicionalmente, un *synset* contiene una breve definición conocida como "*gloss*", y en la mayoría de los casos, una o más sentencias cortas ilustrando el uso de los miembros del *synset*. Formas de palabras con varios significados son representadas en distintos *synsets*. Por lo tanto, cada par forma-significado en WordNet es único. A continuación se muestra un ejemplo de *synset*:

good, right, ripe – (most suitable or right for a particular purpose; "a good time to plant tomatoes"; "the right time to act"; "the time is ripe for great sociological changes")

[2] Un tesauro es un vocabulario controlado y estructurado formalmente, formado por términos que guardan entre sí relaciones semánticas y genéricas: de equivalencia, jerárquicas y asociativas.

2.8.1.2 Relaciones

La relación más frecuente entre *synsets* es la relación hiperonimia - hiponimia (también llamada relación ISA). Vincula *synsets* más generales como {muebles, pieza de muebles} a más específicos como {cama} y {litera}. De este modo WordNet establece la categoría muebles incluyendo cama, que a su vez incluye litera; a la inversa, conceptos como cama y litera conforman la categoría de muebles.

La hiponimia es una relación transitiva: si un sillón es una especie de silla, y si una silla es un tipo de mueble, entonces un sillón es un tipo de mueble. WordNet distingue entre tipos (sustantivos comunes) e instancias (personas específicas, países y entidades geográficas); así, sillón es un tipo de silla, Barack Obama es una instancia de un presidente. Las instancias son siempre nodos hoja (terminal) en sus jerarquías.

Meronimia, la relación parte - todo se mantiene entre *synsets* como {silla} y {espalda, respaldo}, {asiento} y {pata}. Las piezas se heredaron de sus superordinados: si una silla tiene patas, entonces un sillón también tiene patas. Las partes no son heredadas "hacia arriba", ya que pueden ser características sólo de tipos específicos de cosas más que de la clase como un todo: sillas y tipos de sillas tienen patas, pero no todos los tipos de muebles tienen patas.

Los *synsets* verbales se organizan en jerarquías, así: verbos hacia la parte inferior de los árboles (troponimos), expresan modales cada vez más específicos que caracterizan un evento, como en comunicar – hablar - susurrar. La manera específica expresada depende del campo semántico. Los verbos que describen eventos que necesariamente y unidireccionalmente conllevan a otro evento son vinculados de la siguiente manera: {*buy*}-{*pay*}, {*succeed*}-{*try*}, {*show*}-{*see*}, etc.

Los adjetivos se organizan en términos de antonimia. Los pares de antónimos "directos" como *wet* - *dry* y *young* - *old* reflejan la fuerte declaración semántica de sus miembros. Cada uno de estos adjetivos polares a su vez está relacionado con un número de los "semánticamente similares": *dry* está relacionado a *parched*, *arid*, *dessicated* y *bone-dry*, y *wet* relacionado a *soggy*, *watelogged*, etc. Adjetivos semánticamente similares son "antónimos directos".

Sólo hay unos pocos adverbios en WordNet (*hardly*, *mostly*, *really*, etc.) debido a que la mayoría de los adverbios en inglés son derivados directos de adjetivos por medio de afijación morfológica (*surprisingly*, *strangely*, etc.).

2.8.2 OpenThesaurus-es

OpenThesaurus-es[3] es una página web interactiva para el mantenimiento de un tesauro [27]. Un tesauro es un diccionario de sinónimos, por lo tanto, sirve para consultar palabras con un significado igual o parecido. Por ejemplo, buscando "*falso*" se encuentran, entre otras, las palabras "*incorrecto*", "*erróneo*" e "*inexacto*". Todo el mundo puede participar en OpenThesaurus-es y corregir errores o añadir nuevos sinónimos. La función de búsqueda muestra todos los significados en los que aparece una palabra, por ejemplo: *crudo* - *crudo, sin cocer* y como otra entrada para *crudo* - *rudo, bruto*. Si la búsqueda no da resultados se ofrece un enlace donde se puede añadir esa palabra al tesauro.

[3] http://openoffice-es.sourceforge.net/thesaurus/

La base de datos de OpenThesaurus-es solo contiene formas conjugadas, es decir en caso de verbos, solo la forma infinitiva, en caso de sustantivos, solo la forma singular y en caso de adjetivos solo la forma masculina singular, como se muestra en los siguientes ejemplos:

Verbos: *correr* pero no *corre* o *corrió*
Sustantivos: *casa* pero no *casas*
Adjetivos: *rojo* pero no *rojas*

2.8.3 Python

Python[4] es un lenguaje de programación poderoso y fácil de aprender. Cuenta con estructuras de datos eficientes y de alto nivel – listas, diccionarios, tuplas – además de un enfoque simple pero efectivo a la programación orientada a objetos. La elegante sintaxis de Python y su tipado dinámico, junto con su naturaleza interpretada, hacen de éste un lenguaje ideal para *scripting* y desarrollo rápido de aplicaciones en diversas áreas y sobre la mayoría de las plataformas.

El intérprete de Python y la extensa biblioteca estándar están a libre disposición en forma binaria y de código fuente para las principales plataformas desde el sitio web de Python, y puede distribuirse libremente.

El intérprete de Python puede extenderse fácilmente con nuevas funcionalidades y tipos de datos implementados en C o C++ (u otros lenguajes accesibles desde C). Python también puede usarse como un lenguaje de extensiones para aplicaciones personalizables [28].

2.8.4 Clips Pattern

Pattern[5] es un módulo de minería web para el lenguaje de programación Python, se compone de una gran variedad de herramientas utilizadas para minería de datos, procesamiento del lenguaje natural, aprendizaje automático, análisis y visualización de redes.

El módulo **pattern.web** cuenta con herramientas para minería de datos en línea: peticiones asíncronas, una API uniforme para servicios web (Google, Bing, Twitter, Facebook, Wikipedia, Wiktionary, Flickr, RSS), un parser del DOM HTML, funciones de interpretación de etiquetas HTML, un rastreador web, webmail, soporte Unicode.

El módulo **pattern.db** cuenta con envoltorios para bases de datos (SQLite, MySQL), archivos CSV Unicode y datetime de Python. Ofrece una manera más conveniente de trabajar con datos tabulares, por ejemplo, los recuperados con el módulo pattern.web.

El módulo **pattern.en** contiene un etiquetador rápido de *part-of-speech* para el idioma inglés (identifica sustantivos, adjetivos, verbos, etc. en una oración), análisis de sentimientos, herramientas para la conjugación de verbos, singularización y pluralización de sustantivos, y una interfaz de WordNet. Existen contrapartes de este módulo para los idiomas: español, francés, alemán, italiano y holandés; pero en estas no se encuentran implementadas todas las funciones como para el idioma inglés.

El módulo **pattern.search** tiene un sistema de concordancia de patrones similar a las expresiones regulares, se puede utilizar para buscar una cadena por sintaxis (*word function*) o por semántica (*word meaning*).

[4] http://www.python.org/
[5] http://www.clips.ua.ac.be/pattern/

El módulo **pattern.vector** cuenta con herramientas de fácil uso para aprendizaje automático, como son funciones de conteo de palabras, documentos de bolsas de palabras, y un modelo de espacio vectorial para análisis semántico latente y algoritmos de agrupación y clasificación (Naïve Bayes, k-NN, Perceptrón, SVM).

El módulo **pattern.graph** tiene herramientas para el análisis de grafos (camino más corto, centralidad) y visualización de grafos en el navegador. Puede ser usado por ejemplo para el estudio de redes sociales o para modelar relaciones semánticas entre conceptos.

2.8.5 NetworkX

NetworkX[6] es un paquete de software para el lenguaje Python, utilizado para la creación, manipulación y estudio de las estructuras, dinámicas y funciones de redes complejas.
Con NetworkX es posible cargar y almacenar redes en formatos estándar y no estándar, generar varios tipos de redes clásicas y aleatorias, analizar la estructura de la red, construir modelos de red, diseñar nuevos algoritmos de redes, dibujar redes entre otras cosas.
NetworkX cuenta con cuatro clases básicas de grafos: *Graph*, *DiGraph*, *MultiGraph*, *MultiDiGraph*.

Además cuenta con los siguientes algoritmos sobre grafos: *Approxination, Assortativity, Biartite, Blockmodeling, Boundary, Centrality, Chordal, Clique, Clustering, Communities, Components, Connectivity, Cores, Cycles, Directed Acyclic Graphs, Distance Measures, Distance - Regular Graphs, Dominating Sets, Eulerian, Flows, Graphical degree sequence, Hierarchy, Isolates, Isomorphism, Link Analysis, Link Prediction, Matching, Maximal independent set, Minimum Spanning Tree, Operators, Rich Club, Shortest Paths, Simple Paths, Swap, Traversal, Tree, Vitality*.

[6] http://networkx.lanl.gov/

Capítulo 3 Modelos

Para el desarrollo de esta tesis se han propuesto dos modelos, el primero es un modelo basado en aprendizaje supervisado, y el segundo es un modelo de aprendizaje no supervisado; para la implementación de ambos modelos se utilizaron las herramientas Clips Pattern y Network X, además del uso del lenguaje Python; dichos modelos se explican a continuación.

3.1 Modelo 1

Para este modelo se han utilizado como datos de entrenamiento los proporcionados en el corpus correspondiente a la tarea 10 (*Multilingual Semantic Textual Similarity*) del SemEval 2014[7]. Al ser este un modelo basado en aprendizaje supervisado, la tarea principal consiste en encontrar cierto conjunto de características que proporcionen un aprendizaje lo suficientemente bueno para que el clasificador; ya sea máquina de soporte vectorial, Naïve Bayes o red neuronal; pueda realizar un trabajo eficiente. Para este modelo se implementaron 2 variaciones que se explican más adelante y se proponen las siguientes 16 características dadas un par de sentencias S_1 y S_2:

Tabla 3.1. Características empleadas para determinar la Similitud Semántica

Características	
Variación 1	**Variación 2**
TTR[8] mínimo entre S_1 y S_2	TTR promedio de S_1 y S_2
Número mínimo de palabras en S_1 y S_2	Promedio de palabras en S_1 y S_2
Número mínimo de palabras clave[9] en S_1 y S_2	Promedio de palabras clave en S_1 y S_2
Similitud coseno entre S_1 y S_2	
Similitud euclidiana entre S_1 y S_2	
Porcentaje de palabras clave que comparten S_1 y S_2	
Porcentaje de palabras clave que comparten S_1 y S_2 utilizando como *stemmer* el algoritmo de Porter	
Promedio de la intertextualidad entre S_1 y S_2	
Isomorfismo de las representaciones como grafos de S_1 y S_2 (ventana igual a 1)	
Isomorfismo de las representaciones como grafos de S_1 y S_2 (ventana igual a 2)	
Isomorfismo de las representaciones como grafos de S_1 y S_2 (ventana igual a 1, sólo palabras clave)	
Isomorfismo de las representaciones como grafos de S_1 y S_2 (ventana igual a 2, sólo palabras clave)	
Similitud Levenshtein entre S_1 y S_2	
Similitud Dice entre S_1 y S_2	
Distancia coseno entre las características[10] de S_1 y S_2	
Distancia euclidiana entre las características de S_1 y S_2	

[7] http://alt.qcri.org/semeval2014/task10/

[8] El *Type-Token Ratio* es una medida de variación léxica utilizada en varios tipos de análisis lingüísticos, se calcula dividiendo el número de palabras distintas por el número total de palabras, en un texto. Un porcentaje alto de TTR indica una gran cantidad de variación léxica.

[9] Se consideran palabras clave, a todas aquellas que no son consideradas *stopwords* por la herramienta Clips Pattern.

[10] TTR, número de palabras de la sentencia, número de palabras clave de la sentencia.

La diferencia entre las dos variaciones de este modelo es, que para obtener las tres primeras características, en la variación 1 se utiliza el mínimo entre las características de S_1 y las características de S_2; mientras que en la variación 2 se utiliza el promedio de las características de S_1 y S_2.

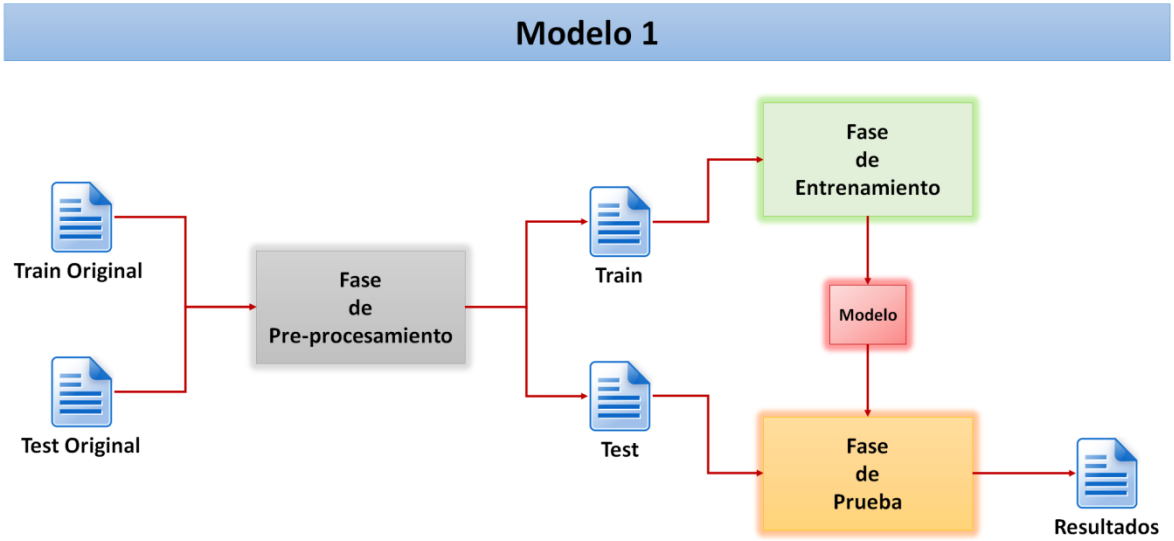

Figura 3.1. Diagrama del Modelo 1

Como se puede ver en la Figura 3.1, este modelo fue implementado en tres fases, la "Fase de Pre-procesamiento", la "Fase de Entrenamiento" (ver Figura 3.2) y la "Fase de Prueba" (ver Figura 3.8). A continuación se explican cada una de estas fases.

3.1.1 Fase de Pre-procesamiento

Esta fase es la encargada de estandarizar los archivos de entrada, sustituyendo los caracteres que no entran en la codificación ANSI por sus correspondientes, además de eliminar símbolos, signos de puntuación y las *stopwords* más comunes de cada idioma, las cuales se presentan en el Anexo 1. En el siguiente algoritmo se muestra el procedimiento que se realiza en esta fase.

1. Se sustituyen los caracteres que no se encuentran en la codificación ANSI.
2. Cada sentencia se convierte a minúsculas.
3. Se eliminan las *stopwords* de cada sentencia, las cuales se presentan en el Anexo 1.
4. Se eliminan los signos de puntuación y símbolos de cada sentencia.
5. Se eliminan las palabras repetidas de cada sentencia.

En la Tabla 3.2 se muestran los símbolos y signos de puntuación que son eliminados en el paso 3 del algoritmo anterior.

Tabla 3.2. Signos y símbolos a eliminar

Signos de Puntuación	Símbolos
.,:; !?¿¡	_"#@()=+-*/| []{}$%~<>^

3.1.2 Fase de Entrenamiento

Como se puede apreciar en la Figura 3.2, la fase de entrenamiento consiste en obtener las características de cada par de sentencias de entrada, con lo que se consigue el "Vector de Características" que junto con su grado de similitud serán utilizados por el clasificador para entrenar y así obtener el modelo utilizado para la fase de prueba.

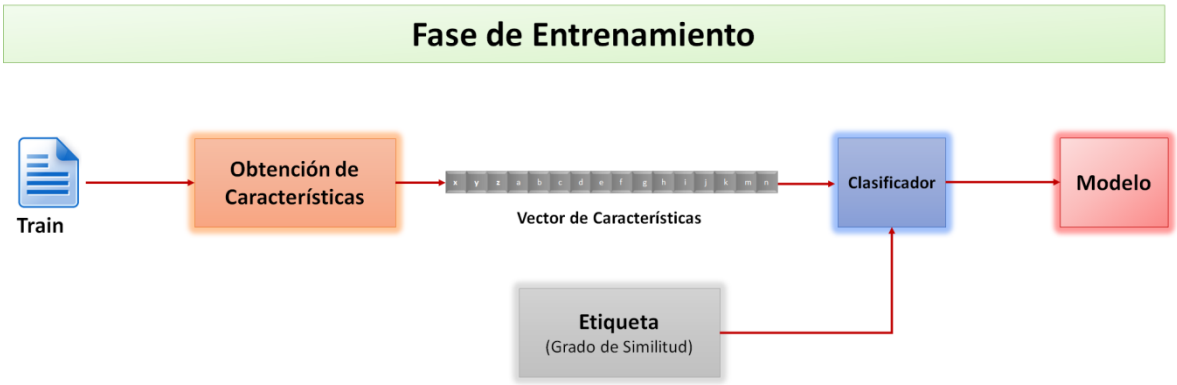

Figura 3.2. Diagrama de la Fase de Entrenamiento

La parte más importante, como se mencionó anteriormente, consiste en la obtención de características. En la Figura 3.7 se muestra el diagrama del proceso de obtención de las características. Este proceso será descrito en 5 pasos.

Paso 1:

Se comienza recibiendo las sentencias S_1 y S_2, después se obtienen tres características propias de cada sentencia, estas características son:

- ✓ *Type-Token Ratio* (TTR)
- ✓ Numero de palabras en la sentencia
- ✓ Numero de palabras clave (no *stopwords*) en la sentencia

Con estas características se forman los vectores "$carS_1$" y "$carS_2$", para las sentencias S_1 y S_2 respectivamente.

Para el cálculo del *Type-Token Ratio* de cada sentencia se empleó la función *ttr()* de la herramienta Clips Pattern, dicha función calcula el TTR de un texto cada *n* sucesivas palabras, esta recibe dos parámetros, un *string* (la sentencia) y un entero como valor de *n*; se empleó un valor de *n* igual a la cantidad de palabras en la sentencia.

Paso 2:
Utilizando las sentencias S_1 y S_2, se calculan las características que comparten. Dichas características son las siguientes:

- ✓ Similitud coseno
- ✓ Similitud euclidiana
- ✓ Porcentaje de palabras clave que comparten
- ✓ Porcentaje de palabras clave que comparten utilizando como *stemmer* el algoritmo de Porter
- ✓ Promedio de la intertextualidad
- ✓ Isomorfismo de sus representaciones como grafos (ventana igual a 1)
- ✓ Isomorfismo de sus representaciones como grafos (ventana igual a 2)
- ✓ Isomorfismo de sus representaciones como grafos (ventana igual a 1, solo palabras clave)
- ✓ Isomorfismo de sus representaciones como grafos (ventana igual a 2, solo palabras clave)
- ✓ Similitud Levenshtein
- ✓ Similitud Dice

Con estas características se forma el "Vector Compartido". Cabe mencionar que las similitudes coseno y euclidiana se obtuvieron conforme indica la documentación de Clips Pattern, con la fórmula *1-distance()*, empleando la función *distance()* que proporciona dicha herramienta.

La representación de cada sentencia por medio de un grafo se llevó a cabo utilizando la herramienta Network X. Cada grafo fue construido de la siguiente manera:

1. Se crea un grafo vacío no dirigido.
2. Se obtienen las palabras de la sentencia, o sólo las palabras claves, dependiendo el caso.
3. Para cada palabra que se obtuvo se agrega una arista que una a esta palabra y a las siguientes *n* palabras, donde *n* es el valor de la ventana.
4. Se remueven los auto ciclos[11].

Como se puede notar, cada palabra distinta en la sentencia se convertirá en un nodo dentro del grafo, así bien los nodos se conectaran con otros nodos si las palabras que representan dichos nodos co-ocurren dentro del valor de la ventana.

En la Figura 3.3 se observa la representación por medio de un grafo con ventana igual a 1 de la sentencia "*A woman and man are dancing in the rain*". En este tipo de representación cada palabra sólo está conectada con su anterior y su siguiente dentro de la sentencia.

[11] Se considera auto ciclo cuando un nodo se encuentra conectado a si mismo por medio de una arista.

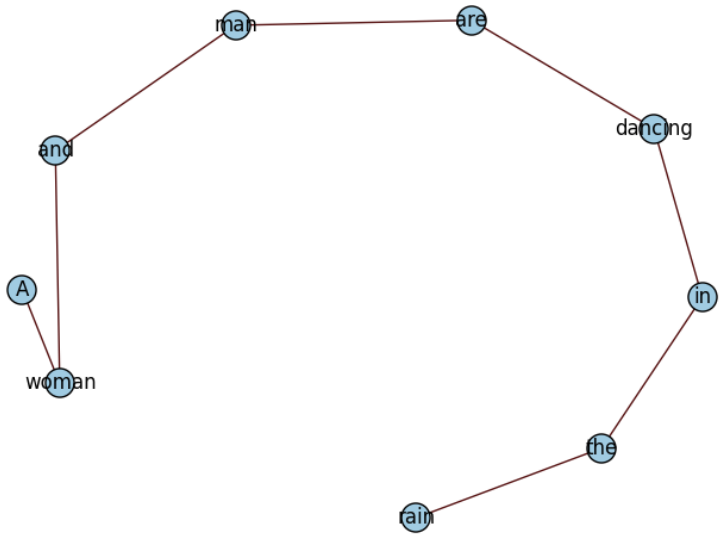

Figura 3.3. Grafo con ventana=1

Al usar un valor de ventana mayor a 1 se obtiene un grafo en donde las palabras no solo se asocian con sus vecinos inmediatos dentro de la sentencia, lo cual puede ayudar en la determinación de la co-ocurrencia de palabras; en la Figura 3.4 se presenta la misma sentencia utilizando una ventana igual a 2 para la construcción del grafo.

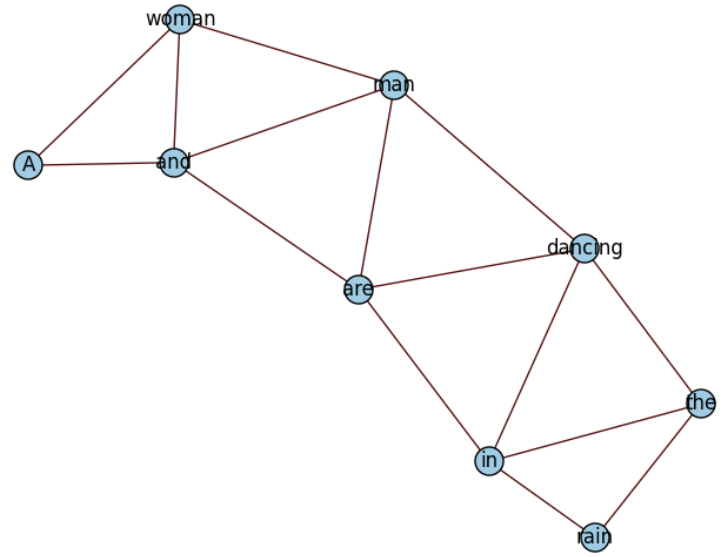

Figura 3.4. Grafo con ventana=2

Cuando únicamente se utilizan las palabras clave para la representación de los grafos, se corre el riesgo de que se obtenga un grafo vacío en el caso de que la sentencia se encuentre formada únicamente por palabras que se encuentran catalogadas como *stopwords* por la herramienta Clips Pattern. Pero al mismo tiempo, se eliminan palabras que no aportan suficiente información en la sentencia. En la Figura 3.5 se aprecia lo recién mencionado, ya que se eliminaron de la sentencia palabras que no

aportan mucha información como es el caso de "*A*", "*in*", "*the*"; pero también se perdieron otras que sí podrían retribuir información como lo es "man".

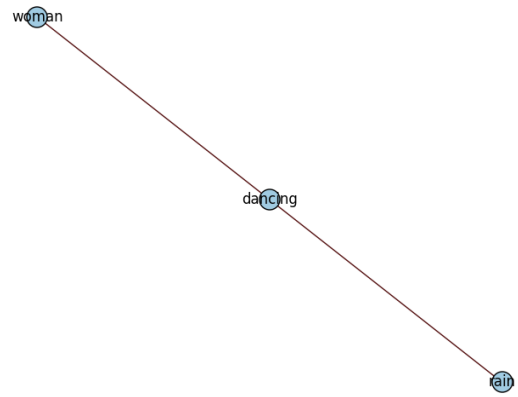

Figura 3.5. Grafo con ventana=1, solo palabras clave

En la Figura 3.6 se muestra la representación de la misma sentencia, ahora usando ventana igual a 2 y solo representando las palabras clave.

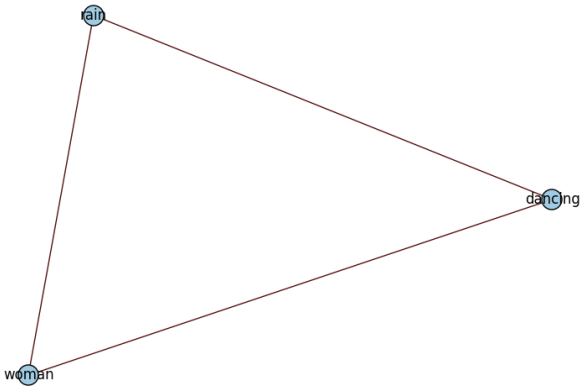

Figura 3.6. Grafo con ventana=2, solo palabras clave

El isomorfismo de las representaciones como grafos de las sentencias S_1 y S_2 se obtiene utilizando la función *could_be_isomorphic()* que proporciona la herramienta Network X, dicha función recibe como parámetros dos grafos no dirigidos y devuelve *True* en el caso de que los grafos pudieran ser isomorfos.

Las similitudes Levenshtein y Dice fueron calculadas con sus funciones correspondientes en la herramienta Clips Pattern.

Paso 3:

Utilizando los vectores $carS_1$ y $carS_2$, que se obtienen en el paso 1, se crea el "Vector de Distancias", este vector se compone de las distancias coseno y euclidiana entre los vectores $carS_1$ y $carS_2$.

Paso 4:
Se "unifican" los vectores $carS_1$ y $carS_2$ para obtener el "Vector Unificado", dicho vector en la variación 1 de este modelo, se obtiene de la siguiente manera:

$$vectorUnificado[i]=min(carS_1[i],carS_2[i])$$

Donde *min(a,b)* es el menor de los valores de *a* y *b*.

Para la variación 2 de este modelo, el "Vector Unificado" se obtiene de la siguiente forma:

$$vectorUnificado[i]=(carS_1[i]+carS_2[i])/2$$

Paso 5:
Se realiza la unión de los vectores Unificado, Compartido y Distancias, para así obtener el "Vector de Características", el cual representa ambas sentencias.

A continuación se muestra en la Figura 3.7 el diagrama del proceso de obtención de características.

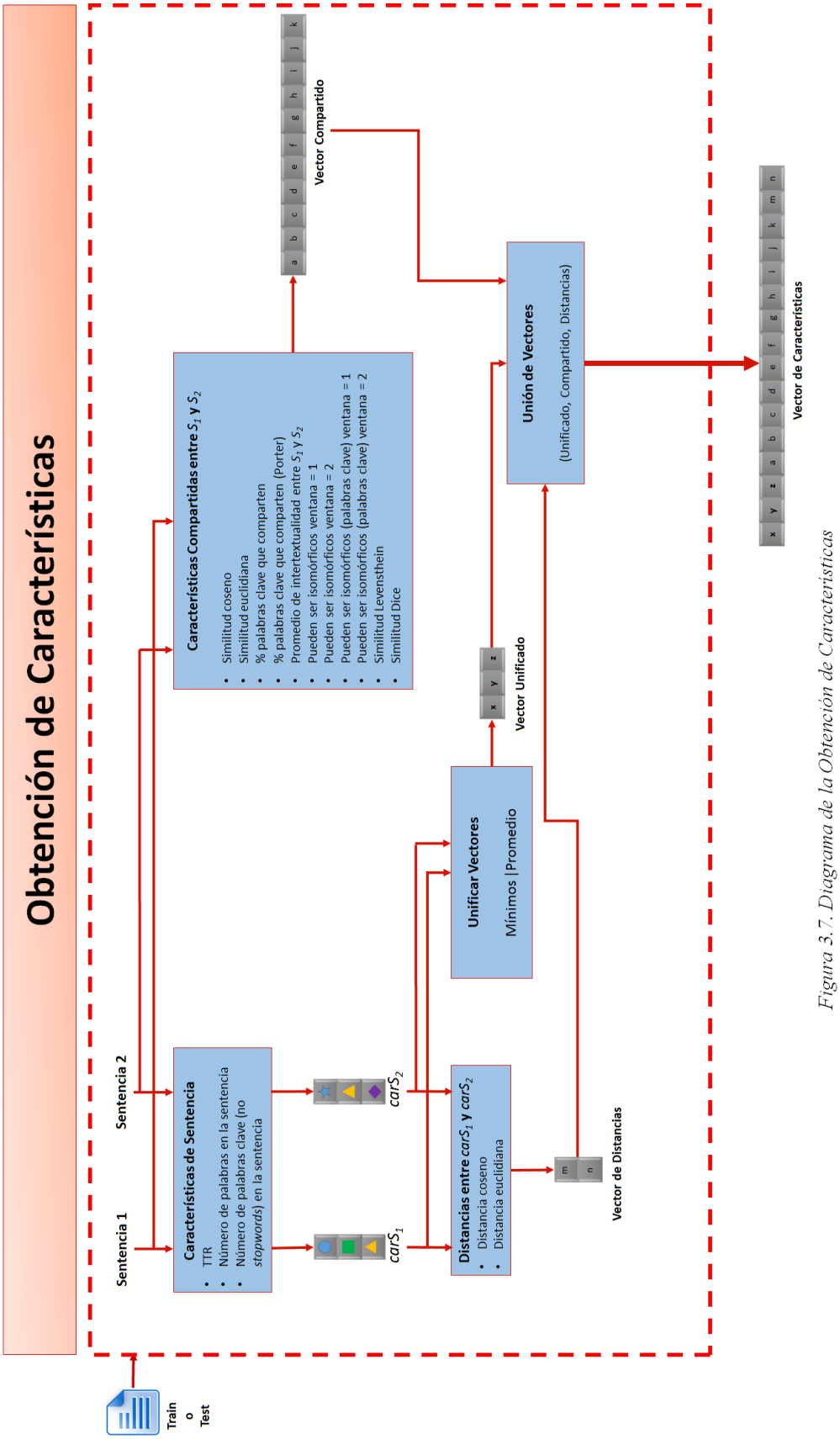

Figura 3.7. Diagrama de la Obtención de Características

3.1.3 Fase de Prueba

Como se puede observar en la Figura 3.8, la fase de prueba consiste en la obtención de características de cada par de sentencias, siguiendo el mismo método que en la fase de entrenamiento, para así obtener el "Vector de Características", el cual será pasado al modelo que se obtuvo por medio del clasificador (Máquina de Soporte Vectorial, Naïve Bayes, Red Neuronal), para que este asocie el vector de características con el grado de similitud que más se le asemeje.

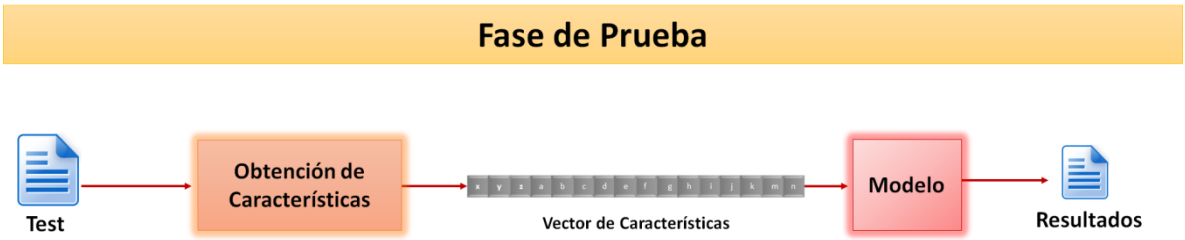

Figura 3.8. Diagrama de la Fase de Prueba

3.2 Modelo 2

El segundo modelo propuesto para este trabajo de tesis, se basa en un algoritmo simple que no requiere entrenamiento, en éste se determina la similitud semántica de dos sentencias por medio de la intertextualidad que existe entre la sentencia que posee el menor número de palabras distintas y la reconstrucción de está utilizando la sentencia con mayor número de palabras distintas.

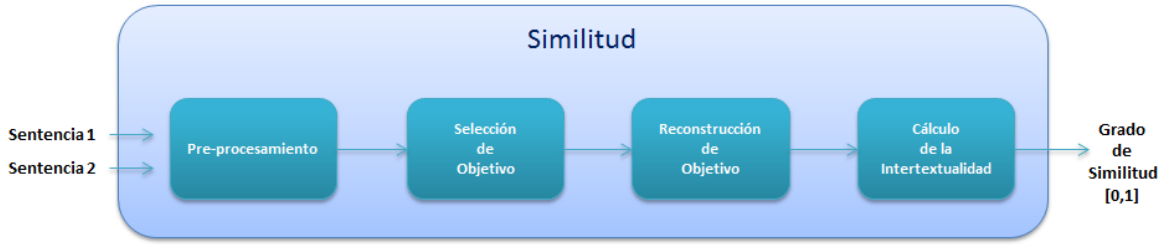

Figura 3.9. Obtención de la Similitud Semántica

Como se puede ver en la Figura 3.9, este modelo está formado por cuatro etapas, tras las que, una vez terminado el proceso se obtendrá un grado de similitud representado por un valor entre cero y uno. Donde, un valor igual a cero significa que las sentencias son totalmente distintas y un valor igual a uno que ambas sentencias representan el mismo concepto. A continuación se describen las cuatro etapas de este modelo.

3.2.1 Pre-procesamiento

La primera etapa comienza convirtiendo ambas sentencias a su forma en minúsculas, seguido de la eliminación de símbolos y signos de puntuación (ver Tabla 3.2). Con esto una sentencia como "*¿El tic-tac del reloj?...*", cambia a la siguiente manera "*el tictac del reloj*". Posteriormente se eliminan las palabras repetidas dentro de cada sentencia además de las *stopwords* correspondientes al idioma de la sentencia, dichas *stopwords* son las presentadas en el Anexo 1.

3.2.2 Selección de Objetivo

Esta etapa consiste en determinar cuál de las dos sentencias se intentará reconstruir, se propone que dicha sentencia sea la de menor longitud en términos de palabras distintas, ya que en la etapa de pre-procesamiento se eliminan las palabras repetidas. La sentencia que no se elige como objetivo, la de mayor longitud, servirá para reconstruir la sentencia objetivo.

3.2.3 Reconstrucción de Objetivo

Usando la sentencia de mayor longitud se intentará reconstruir la sentencia objetivo aplicando el siguiente algoritmo:

```
for w in Sm:
        if(w in So):
                Sr = Sr + w
        else:
                if(existeSinónimo(w,So)):
                        Sr = Sr + sinónimo(w,So)
```

Dónde:
w es una palabra que pertenece a *Sm*
Sm representa la sentencia de mayor longitud
So representa la sentencia objetivo
Sr representa la sentencia resultado de la reconstrucción
La función *existeSinónimo()* determina si está presente un sinónimo de *w* en *So*
La función *sinónimo()* devuelve el sinónimo de *w* presente en *So*

Para determinar la sinonimia entre las palabras se usó, WordNet para el idioma inglés, y para el idioma español se realizó una adaptación del OpenThesaurus-es para que sus sinónimos fueran accesibles desde el lenguaje Python.

3.2.4 Cálculo de la Intertextualidad

Bajo la premisa de que, a mayor cantidad de palabras compartidas en cada sentencia mayor es la probabilidad de que representen la misma idea, se utiliza la intertextualidad para medir cuanto de la sentencia objetivo se pudo reconstruir en la sentencia resultado. El cálculo de la intertextualidad se llevó a cabo utilizando la función *intertextuality()* que proporciona la herramienta Clips Pattern. Dando el resultado de esta operación como el grado de similitud semántica entre ambas sentencias.

Capítulo 4 Análisis de Resultados

Como se ha mencionado anteriormente, para el desarrollo y prueba de ambos modelos planteados, se han utilizado los corpus proporcionados en la tarea 10 (*Multilingual Semantic Textual Similarity*) de la conferencia SemEval 2014.

4.1 Conjunto de Datos

Se dispone de un corpus de entrenamiento para el idioma inglés conformado por 6 627 pares de sentencias etiquetadas con valores enteros entre cero y cinco representando el grado de similitud entre cada par de sentencias (Tabla 4.1). Para el entrenamiento de ambos idiomas se utilizaron los datos del corpus del idioma inglés.

Tabla 4.1. Composición del corpus de entrenamiento

Corpus de Entrenamiento			
	Total de Sentencias	**Idioma de Entrenamiento**	**Total de Sentencias de Entrenamiento**
Inglés	6 627	Inglés	6 627
Español	0	Inglés	6 627

El corpus de prueba para el idioma inglés lo forman 3 000 pares de sentencias. No obstante para el idioma español sólo se cuenta con un pequeño corpus de 65 pares de sentencias, el cual solo se usó para realizar las pruebas (Tabla 4.2).

Tabla 4.2. Composición del corpus de prueba

Corpus de Entrenamiento	
Idioma	**Total de sentencias**
Inglés	3 000
Español	65

4.2 Primer Modelo

Durante las pruebas de este modelo, se emplearon seis clasificadores para cada variación del mismo. Se utilizaron los siguientes clasificadores que proporciona la herramienta Clips Pattern:

- Máquina de Soporte Vectorial utilizando un *kernel* lineal
- Máquina de Soporte Vectorial utilizando un *kernel* polinomial de segundo grado
- Máquina de Soporte Vectorial utilizando un *kernel* polinomial de tercer grado
- Máquina de Soporte Vectorial utilizando un *kernel* radial
- Naïve Bayes
- Perceptrón Simple

A continuación, se muestran los resultados obtenidos de la implementación de este modelo en sus dos variaciones, tanto para el idioma inglés como para el idioma español.

4.2.1 Idioma Inglés

En la Tabla 4.3, se muestran los resultados de precisión que obtuvo el modelo 1 en su variación 1(mínimos). En esta se puede apreciar que el mejor resultado se logró utilizando como clasificador una máquina de soporte vectorial con *kernel* lineal, obteniendo de esta manera un total de 1,029 aciertos los cuales representan un 34.30% de precisión. Cabe mencionar que el peor desempeño fue por parte de las máquinas de soporte vectorial con *kernels* polinomial de segundo grado, polinomial de tercer grado y radial; ya que estas no mostraban variación alguna en su clasificación, dando siempre un grado de similitud de 4.

Tabla 4.3. Resultados de Precisión, idioma inglés, variación 1

Inglés, Variación 1		
Clasificador	Total de Aciertos	Porcentaje
SVM Kernel Lineal	1029	34.30%
SVM Kernel Polinomial 2° grado	503	16.76%
SVM Kernel Polinomial 3° grado	503	16.76%
SVM Kernel Radial	503	16.76%
Naïve Bayes	892	29.73%
Perceptrón Simple	741	24.70%

En seguida se muestra, en la Tabla 4.4, el porcentaje de precisión obtenido en el modelo 1 en su segunda variación (promedio). De la misma forma que en la variación 1, el peor desempeño lo tienen las máquinas de soporte vectorial con *kernels* polinomial de segundo grado, polinomial de tercer grado y radial, mostrando el mismo comportamiento al asignar siempre un grado de similitud de 4. Por otra parte, el mejor resultado es logrado por medio de la máquina de soporte vectorial con *kernel* lineal, mostrando una precisión de 35.16% con 1,055 aciertos.

Tabla 4.4. Resultados de Precisión, idioma inglés, variación 2

Inglés, Variación 2		
Clasificador	Total de Aciertos	Porcentaje
SVM Kernel Lineal	1055	35.16%
SVM Kernel Polinomial 2° grado	503	16.76%
SVM Kernel Polinomial 3° grado	503	16.76%
SVM Kernel Radial	503	16.76%
Naïve Bayes	892	29.73%
Perceptrón Simple	830	27.66%

Al comparar los resultados de este modelo para el idioma inglés, se observa que la variación 2 (promedio) conlleva a una mayor precisión.

4.2.2 Idioma Español

Como se aprecia en la Tabla 4.5, al igual que para el idioma inglés, el mejor resultado en la variación 1 (mínimos) lo obtiene la máquina de soporte vectorial con *kernel* lineal, dando un total de 42 aciertos (64.61%); mientras que los otros tipos de máquina de soporte vectorial mantienen el mismo comportamiento, asignando siempre un grado de similitud de 4.

Tabla 4.5. Resultados de Precisión, idioma español, variación 1

Español, Variación 1		
Clasificador	Total de Aciertos	Porcentaje
SVM Kernel Lineal	42	64.61%
SVM Kernel Polinomial 2° grado	0	0%
SVM Kernel Polinomial 3° grado	0	0%
SVM Kernel Radial	0	0%
Naïve Bayes	9	13.84%
Perceptrón Simple	1	1.53%

Los resultados de la variación 2 (promedio) para el idioma español son expuestos en la Tabla 4.6, donde el mejor porcentaje fue de 52.30%, con un total de 34 aciertos por parte de la máquina de soporte vectorial con *kernel* lineal.

Tabla 4.6. Resultados de Precisión, idioma español, variación 2

Español, Variación 2		
Clasificador	Total de Aciertos	Porcentaje
SVM Kernel Lineal	34	52.3%
SVM Kernel Polinomial 2° grado	0	0%
SVM Kernel Polinomial 3° grado	0	0%
SVM Kernel Radial	0	0%
Naïve Bayes	9	13.84%
Perceptrón Simple	7	10.76%

Al contrario que en el idioma inglés, el mejor desempeño se logró en la variación 1 (mínimos), y a pesar de que todo el entrenamiento se realizó con un corpus para el idioma inglés, la mayor precisión de este modelo se observa en el idioma español.

4.3 Segundo Modelo

En este apartado se presentan los resultados obtenidos por el modelo 2. Para realizar una comparación entre los aciertos, tomando en cuenta que dicho modelo representa la similitud semántica con un valor en el rango [0,1], se ha multiplicado el valor que retorna por cinco para después sólo conservar la parte entera; la cual se comparó con los resultados proporcionados en los corpus de prueba para el idioma inglés y español.

Como se puede ver en la Tabla 4.7, este modelo presenta una mayor tasa de aciertos con respecto al modelo 1. El porcentaje obtenido para el idioma inglés es de 37.86%, con un total de 1,136 aciertos y para el idioma español se obtuvieron 46 aciertos para un porcentaje de 70.76%.

Tabla 4.7. Resultados de Precisión para el Modelo 2

Idioma	Relación de Aciertos	Porcentaje
Inglés	1136 / 3000	37.86 %
Español	46 / 65	70.76 %

Conclusiones y Recomendaciones

Durante el desarrollo del presente trabajo de tesis se cumplieron todos y cada uno de los objetivos planteados. Se desarrollaron dos modelos para medir la similitud semántica, en base a los experimentos realizados con ambos modelos se llegó a las siguientes conclusiones:

1. La utilización de un *kernel* lineal en una máquina de soporte vectorial proporciona un mejor desempeño, ya que de esta manera se observaron los mejores resultados en el modelo de aprendizaje supervisado.
2. Las características propuestas para el entrenamiento del modelo 1 presentan un buen comportamiento para ambos idiomas.
3. Emplear isomorfismo de grafos no proporcionó una mayor ventaja con respecto a otras características, sin embargo, es un buen discriminante cuando ambas sentencias muestran una total equivalencia.
4. Los mejores resultados entre ambos modelos fueron por parte del modelo 2, tanto en el idioma inglés como español.
5. Ambos modelos presentan un comportamiento estable para los dos idiomas en los que se probaron.

Como recomendaciones para trabajos futuros se realizan las siguientes sugerencias:

1. Realizar un modelo basado en grafos en el que se incluyan los sinónimos de las palabras, para después buscar el mayor sub-grafo común entre las sentencias.
2. Utilizar la medida de similitud propuesta en el Modelo 2 como una característica para un nuevo modelo basado en aprendizaje supervisado.
3. Incluir diccionarios de abreviaturas, siglas y modismos para normalizar las sentencias en caso de que contengan lo antes mencionado.

Anexo 1: Stopwords

Se consideran como *stopwords* las palabras que al aparecer en todo tipo de textos no retribuyen suficiente información, la mayor parte de estas palabras son conjunciones, pronombres, sustantivos y algunos verbos. En este anexo se presentan las *stopwords* más utilizadas tanto para el idioma inglés como para el español.

Para su obtención, se construyó un grafo con ventana igual a 1 para cada idioma, utilizando todas las sentencias del corpus de entrenamiento proporcionado en la tarea *Multilingual Semantic Textual Similarity: Subtask English* del SemEval 2014 para el idioma inglés, y para el idioma español se utilizaron los datos de prueba proporcionados en la tarea *Multilingual Semantic Textual Similarity: Subtask Spanish* también del SemEval 2014.

Una vez construido el grafo se aplicó la centralidad de grado (*degree centrality*) sobre este, dicha centralidad corresponde al número de enlaces que posee un nodo con respecto a los demás. Con lo cual a mayor cantidad de enlaces en un nodo, la palabra representada por ese nodo es utilizada en una mayor cantidad de textos, lo que le resta importancia convirtiéndola en *stopword*.

En la siguiente lista se muestran las *stopwords* que tienen una incidencia de más del 1% en los textos de entrada para el idioma inglés.

> *the, a, of, in, and, to, or, is, for, on, with, that, by, as, at, from, an, was, are, said, be, has, it, this, its, not, after, us, which, will, have, his, were, but, into, over, who, new, up, two, more, he, some, had, i, also, about, their, something, one, we, no, out, can, man, against, they, you, would, people, being, police, all, s, may*

Por último, se muestran a continuación las *stopwords* con una incidencia mayor al 1% para el idioma español.

> *de, y, la, en, el, que, a, del, los, un, se, por, es, para, una, con, las, al, o, su, como, más, no, entre, ha, fue, sus, desde, son, este, sobre, está, también, según*

Bibliografía

[1] N. Malandrakis, E. Iosif, V. Prokopi, A. Potamianos, S. Narayanan. (2013). Lexical, String and Affective Feature Fusion for Sentence-Level Semantic Similarity Estimation. [Online]. Recuperado el 12 de Marzo del 2015 de: http://clic.cimec.unitn.it/starsem2013-program/52_Paper.pdf

[2] M. Heilman, N. Madnani. (2013). Domain Adaptation and Stacking for Text Similarity. [Online]. Recuperado el 12 de Marzo del 2015 de: http://clic.cimec.unitn.it/starsem2013-program/51_Paper.pdf

[3] J. Xu, Q. Lu. (2013). Computing Semantic Textual Similarity using Overlapped Senses. [Online]. Recuperado el 12 de Marzo del 2015 de: http://clic.cimec.unitn.it/starsem2013-program/49_Paper.pdf

[4] H. Ziak, R. Kern. (2013). Semantic Text Similarity by use of Knowledge Bases. [Online]. Recuperado el 12 de Marzo del 2015 de: http://clic.cimec.unitn.it/starsem2013-program/59_Paper.pdf

[5] A. Chávez, A. Fernández, H. Dávila, Y. Gutiérrez, A. Collazo, J. Abreu, A. Montoyo, R. Muñoz. (2013).Textual Similarity Based on Lexical-Semantic Features. [Online]. Recuperado el 12 de Marzo del 2015 de: http://clic.cimec.unitn.it/starsem2013-program/53_Paper.pdf

[6] Datamining (Minería de Datos). [Online]. Recuperado el 16 de Abril del 2015 de: http://www.sinnexus.com/business_intelligence/datamining.aspx

[7] Minería de textos y sus aplicaciones. [Online]. Recuperado el 17 de Abril del 2015 de:http://www.semanticwebbuilder.org.mx/es/swb/Mineria_de_textos_y_sus_aplicaciones

[8] Lingüística Computacional. [Online]. Recuperado el 17 de Abril del 2015 de: http://www.ecured.cu/index.php/Ling%C3%BC%C3%ADstica_computacional

[9] Gelbukh, A., Sidorov, G. (2006). Procesamiento Automático del Español con enfoque en recursos léxicos grandes. (p./pp. 63-69). Instituto Politecnico Nacional, Mexico, DF. ISBN: 970-36-0264-9.

[10] Mejía Sánchez-Bermejo, Antonio. (2013). Similitud semántica entre conceptos de Wikipedia. [Online]. Recuperado el 14 de Marzo del 2015 de: http://e-archivo.uc3m.es/handle/10016/17170

[11] Manning, D. C., Raghavan, P., Schütze, H., (2009). An Introduction to Information Retrieval (p. 121). Cambridge UP.

[12] Deza, E., Deza, M., (2009). Encyclopedia of Distances (p. 94). Springer.

[13] Black, P. E. (2008). Levenshtein distance. Dictionary of Algorithms and Data Structures, U.S. National Institute of Standards and Technology.

[14] Murguía, M., Villaseñor, J. L. (2003). Estimating the effect of the similarity coefficient and the cluster algorithm on biogeographic classifications. (p./pp. 415-421).

[15] Allen, M., (1998). Data Structures and Problems Solving using Java. (p. 353). Pearson.

[16] Jiménez, José. (2009). Matemáticas para la computación. (p. 303). Alfaomega.

[17] Polanco. X. (2008). Análisis de redes: introducción. [Online]. Recuperado el 22 de Enero del 2015 de: https://halshs.archives-ouvertes.fr/hal-00218397/document

[18] Borgatti, S. P. (2005). Centrality and network flow. (p./pp. 55–71).

[19] Hernández M., Gómez J. (2013). Aplicaciones de Procesamiento de Lenguaje Natural. [Online]. Recuperado el 20 de Abril de 2015 de: http://rua.ua.es/dspace/bitstream/10045/33514/1/2013_Hernandez_Gomez_RevPolitec.pdf

[20] Betancourt A. G. (2005). Las Máquinas de Soporte Vectorial (SVMs). [Online]. Recuperado el 27 de Marzo del 2015 de: http://dialnet.unirioja.es/descarga/articulo/4838384.pdf

[21] Maldonado, S., Weber, R. (2012). Modelos de Selección de Atributos para Support Vector Machines. [Online]. Recuperado el 27 de Marzo del 2015 de: http://www.dii.uchile.cl/~ris/RISXXVI/maldonado1.pdf

[22] Clasificador Naïve Bayes. [Online]. Recuperado el 28 de Marzo del 2015 de: http://naivebayes.blogspot.mx/

[23] Redes Neuronales. [Online]. Recuperado el 29 de Marzo del 2015 de: http://avellano.usal.es/~lalonso/RNA/

[24] Inteligencia Artificial. [Online]. Recuperado el 29 de Marzo del 2015 de: http://www2.ulpgc.es/hege/almacen/download/38/38584/practica_ia_2.pdf

[25] Perceptrón simple. [Online]. Recuperado el 29 de Marzo del 2015 de: http://www.lab.inf.uc3m.es/~a0080630/redes-de-neuronas/perceptron-simple.html

[26] WordNet a lexical database for English. [Online]. Recuperado el 26 de Febrero del 2015 de: http://wordnet.princeton.edu/

[27] OpenThesaurus-es FAQ. [Online]. Recuperado el 4 de Marzo del 2015 de: http://openoffice-es.sourceforge.net/thesaurus/faq.php

[28] Van Rossum, G. (2009). El tutorial de Python. [Online]. Recuperado el 20 de Abril del 2015 de: http://docs.python.org.ar/tutorial/pdfs/TutorialPython2.pdf