DESARROLLO DE UN APLICATIVO PARA EL APOYO EN LA CONSTRUCCIÓN DE MARCOS TEÓRICOS A PARTIR DE MINERÍA DE TEXTO.

R. O. Peña – Pérez
P. A. Briceño – Rivera
H. A Mora – Paz
D. Mayorca
Corporación Universitaria Autónoma de Nariño
Pasto Colombia

Resumen — La necesidad de mejorar los sistemas informáticos para comprender y minimizar el tiempo en diversas tareas hace que se creen nuevos conceptos, uno de ellos es el procesamiento del lenguaje natural (NLP), el cual tiene como objetivo que una maquina pueda conocer, interpretar y crear lenguaje humano coherente. Con ello se puede realizar un apoyo en diversas tareas de nivel investigativo como lo es la creación de un marco teórico. Para solventar este apoyo en esta investigación se aporta con la creación de un aplicativo donde se realiza un compendio de algoritmos, uno de ellos para modelado de temas como lo es el algoritmo LDA, al igual que se hace uso de bibliotecas como spaCy y nltk, que son bibliotecas reconocidas para NLP. Con este compendio de algoritmos se da como resultado la comprensión, análisis y una reducción en el tiempo de lectura de varios artículos, con ello se cumple la finalidad de apoyar en la construcción de un marco teórico.

Palabras Claves – *procesamiento de lenguaje natural (NLP), maching learning, LDA, spaCy, nltk.*

I. INTRODUCCIÓN

El desarrollo en el campo de procesamiento de lenguaje natural y aprendizaje automático permite que los sistemas puedan adquirir conocimiento a partir de un conjunto de datos y mejoren con la experiencia, algunos de estos sistemas son: la robótica y vehículos autónomos, las redes neuronales, los motores de búsqueda, el procesamiento del lenguaje natural, detección de rostros y anti-spam, como algunos que se destacan en el campo de la inteligencia artificial (IA).

En un futuro cercano, asistentes personales inteligentes como Siri y Alexa, los automóviles autónomos y los algoritmos de diagnóstico de enfermedades serán solo herramientas de una Inteligencia Artificial más avanzada.

Actualmente se encuentra en auge el desarrollo de herramientas de inteligencia artificial basadas en lenguaje natural para el procesamiento de texto en línea en blogs y noticias como *wordsmith* [1].

Su principal uso se encuentra en lograr clasificar textos electrónicos que proporcionen un marco que permita la conceptualización de teorías sobre cómo analizar contenido para la adquisición de conocimiento.

Por lo general, estas nuevas herramientas tecnológicas de uso en internet se centran en difundir información a un público más amplio, dado que los usuarios que usan internet no solo consumen información si no también generan contenido en múltiples fuentes.

Estas técnicas de procesamiento de lenguaje natural proponen formas más efectivas para llegar a integrar el conocimiento con los estudiantes, principalmente en áreas de preferencia investigativa en las que faltan recursos básicos.

Por tanto, la difusión del conocimiento en la investigación educativa promueve el papel de la comunidad científica para producir fuentes de datos y publicaciones que faciliten explicar nuevas ideas. Si bien, el procesamiento de lenguaje natural puede generar información más adecuada para comunicar la ciencia a investigadores, se podría potencializar la creación de nuevas formas de estudio en los que el conocimiento y el contenido novedoso se utilizan para conectarse con nuevas formas de investigación.

En algunos casos la necesidad de investigadores es encontrar la representación de información de un espacio de datos original en un esquema similar a un marco teórico que permita evitar información irrelevante, errónea o redundante inmersa en documentos o sitios contenedores de texto. Idealmente se busca minimizar la confusión de resultados para presentar una mejor lectura de la información reduciendo el tiempo operacional, el coste computacional de procesamiento y lograr mejorar la calidad del informe presentado al usuario.

Para esto hay que tener en cuenta que es posible razonar acerca de la información generada, ya sea en términos de la estructura semántica o del perfil de la

Con formato: Fuente: Cursiva, No revisar la ortografía ni la gramática

aplicación que crea la línea de palabras que fueron declaradas por el procesamiento de lenguaje como un problema de optimización.

En estas estructuras generadas por el análisis se podrían encontrar dimensiones altamente correlacionadas con otras considerándolas como redundantes, cuando existen características que no estén correlacionadas con otras o que estén presentes con mayores variaciones en la medición se las puede considerar ~~según la estructura de los datos~~ como características erróneas según se presente la estructura de los datos.

Dado que el aprendizaje es un proceso complejo con variaciones sustanciales basadas en los antecedentes, la cultura, la historia, la motivación, el dominio del oficio o la información disponible, es necesario desarrollar aplicativos que incluyan aprendizaje automático, reconocimiento de patrones y sistemas de recomendación por mencionar algunos, como alternativas que emergen para facilitar las tareas de las personas. Por lo tanto existen razones teóricas y prácticas para procesar información a cantidades que puedan ser más fáciles de manipular e interpretar por el usuario [2][1].

II. MARCO TEÓRICO

A. *Ciencias de la Computación.*

El Desarrollo de Software y la Ingeniería de Software hacen parte de Las Ciencias de la Computación, pero sus definiciones son distintas. El desarrollador se limita a crear programas dentro de la informática~~, "El objetivo de la ingeniería de software sería diseñar cómputos específicos para lograr objetivos prácticos"~~ [3], mientras el Científico se enfatiza en probar hipótesis para así, de alguna manera ampliar el conocimiento intentando métodos casi imposibles de realizar.

La Computación Científica se refiere a las prácticas destinadas a modelar, plantear experimentos para validar teorías científicas sirviéndose de medios computacionales, mientras La Ciencia de la Computación descifra las propiedades de programas que se emplean para implementar aplicaciones de software como lo son el desempeño de las máquinas en los que corren juegos de video y luego ~~utilizar~~ optimizar ~~el~~ este desempeño de ~~las~~ GPU en experimentos de alto coste computacional como PNL (*Model GPT2*).

Las Ciencias de la Computación no detienen su desarrollo y recaen en los Lenguajes de Programación como las herramientas para los distintos niveles de abstracción y complejidad de cada teoría planteada.

B. *Procesamiento de Lenguaje Natural*

El Procesamiento del lenguaje natural ha existido durante más de 50 años y surgió del campo de la lingüística con el aparecimiento de los ordenadores. El Procesamiento del Lenguaje Natural o *NLP* en sus siglas en inglés, realiza un acercamiento a la interpretación del lenguaje natural haciendo así que las computadoras se acerquen a comprender e involucren lenguaje humano de una manera inteligente, logrando llegar a realizar resúmenes de grandes textos, extracción de relaciones, traducción automática, desarrollo de chat bots, etc.

La lingüística computacional es el estudio moderno de la lingüística utilizando las herramientas de la informática. Comprender el lenguaje natural requiere una gran cantidad de conocimiento sobre morfología, sintaxis, semántica y pragmática, así como conocimientos generales sobre el mundo. PNL ya está migrando a conceptos nuevos como NLG que se define como Generación de lenguaje Natural [6].

Otra definición que toma una vertiente interesante es NLU que ~~S~~son las siglas en inglés de "*Natural Language Understanding*" [7], y es una rama de NLP. Su fin es capacitar a un sistema tecnológico para comprender el significado e intención que encierra una frase.

C. *LDA*

Dentro de NLP existen servicios para acercarnos a ese lenguaje humano, como por ejemplo la *"Asignación de Direccionamiento Latente" o "LDA"* [8], en sus siglas en inglés. LDA es un modelo que asigna palabras a diversos temas que se descubren en los documentos.

Este funciona calculando la probabilidad de que una palabra pertenezca a un tema. LDA es muy bueno cuando se quiere descubrir el o los temas que contiene un documento sin haber visto su contenido, haciendo así casi un resumen automático.

Se puede crear un resumen por extracción o abstracción. Un enfoque basado en la extracción detectará los segmentos más importantes del texto de entrada, generalmente oraciones, y los usará para crear un resumen.

Además del problema de decidir la relevancia de cada oración, un resumen basado en extracción tiene que lidiar con la coherencia. Por ejemplo, una oración en el resumen

Con formato: Fuente: No revisar la ortografía ni la gramática

Con formato: Fuente: Sin Negrita, Cursiva, Español (Colombia), No revisar la ortografía ni la gramática

Código de campo cambiado

Con formato: Fuente: Sin Negrita, Español (Colombia), No revisar la ortografía ni la gramática

Con formato: Fuente: Sin Negrita, Cursiva, Español (Colombia), No revisar la ortografía ni la gramática

puede referirse a elementos que no están en el resumen, este es un fenómeno conocido como ***anáfora colgante*** [9], (enriquece el significado).

Al acercar cada vez más la computación al lenguaje humano, se podría mejorar el aprendizaje reduciendo el tiempo de investigación e interpretación de diferentes documentos. El algoritmo LDA posee caracteristicas de creación de oraciones, resúmenes, división de temas, entre otros, generados automáticamente por un sistema computacional ~~ya es un sueño cumplido~~.

III. OBJETIVOS Y METODOLOGÍA

A. *Objetivos.*

Objetivo general. Crear un aplicativo para el apoyo en la construcción de marcos teóricos de artículos, a partir de procesamiento de lenguaje natural.

Objetivos específicos: Definir los requerimientos funcionales del aplicativo necesarios para apoyar el proceso de construcción de un marco teórico mediante análisis sintáctico y lingüístico.

Implementar algoritmos computacionales que permitan la morfología (conocimiento sobre los componentes significativos de las palabras), lingüística y la construcción de un marco teórico a partir de técnicas de Lenguaje Natural.

Diseñar una interfaz gráfica que permitan la gestión de análisis y visualización de diagramas que facilite la interpretación de los resultados obtenidos.

Evaluar funcionamiento de la propuesta metodológica mediante una prueba piloto con artículos en el área de la Neurociencia Cognitiva.

B. *Metodología.*

Este estudio se inscribe en el paradigma positivista, enfoque empírico analítico y pertenece a un tipo de investigación aplicada.

Paradigma. Se podría afirmar que la integración del conocimiento como puente entre las universidades y los recién graduados en el campo laboral fomenta el crecimiento y las habilidades de los investigadores que también emergen fuera del ámbito académico, lo que permite la apertura de los grupos de investigación al crowdsourcing y las agencias de investigación públicas.

Por ende, este trabajo se inclina a un paradigma Positivista, dado el caso que el interés de su dimensión es explicar el fenómeno de comportamiento en la información textual para controlar las fases de construcción de marcos teóricos y generar un análisis riguroso de un modelo más óptimo y representativo de texto.

Figura 1. Descripción Paradigma Positivista. Fuente: Producción propia.

Enfoque. La siguiente investigación es de carácter Empírico Analítico por el análisis a la problemática planteada y por describir de manera objetiva las causas y efectos que justifican esta investigación. Se basa en la experiencia propia y en el análisis de interesados en el campo de la investigación dentro de la Universidad.

Tipo de investigación. El desarrollo de la investigación como la del presente documento, da como resultado el apoyo a un problema concreto a través de conocimientos específicos, lo cual clasifica esta investigación como una investigación aplicada, contribuyendo al conocimiento y al incremento de la comprensión en diversos temas. Esto se verá reflejado en las investigaciones posteriores tanto de estudiantes como investigadores independientes. Por ende, el presente estudio contribuye a una disminución de tiempo en la lectura, creación y extracción de temas.

IV. DESARROLLO

Para el desarrollo de la siguiente investigación se identifica la estructura del artículo, se hace uso de las bibliotecas spaCy y nltk, al igual que se utiliza el algoritmo LDA, por último, se crea un algoritmo para generar un grafo basado en tesauros descubriendo el hiperónimo de sustantivos más frecuentes en el texto de un artículo.

Con formato: Fuente: Cursiva, Español (Colombia), No revisar la ortografía ni la gramática

Con formato: Fuente: Sin Negrita, Sin Cursiva, Español (Colombia), No revisar la ortografía ni la gramática

Con formato: Fuente: Sin Negrita

A. *Identificación estructural del artículo*

El análisis de texto sugiere que los lectores forman una representación mental de la estructura de un documento en el papel que facilita la lectura. Otros estudios afirman que el proceso de aprendizaje en el cerebro no sucede en la lectura sin embargo es una necesidad. La presente investigación examina este concepto de manera empírica para artículos académicos con el fin de hacer recomendaciones.

Algunos estudios demuestran que los lectores de revistas experimentados sí poseen una representación genérica y pueden usarla para organizar fragmentos aislados de texto.

Los lectores de artículos académicos poseen alguna forma de representación mental de la estructura típica de un texto que les permite predecir la ubicación de un párrafo. Esta representación se la puede determinar con modelos estructurales de organización normalizados como lo son: Introducción, Material y métodos, Resultados y Discusión o por sus siglas en inglés "IMRaD". El segundo modelo está compuesto por: Resumen (Abstract), Introducción, Materiales y métodos, Resultados y Discusión (AIMRAD). Por último, el modelo que está formado por: Introducción, Resultados, Discusión, Materiales y métodos (IRDaM).

En el modelo estructural AIMRAD también se halla una subestructura dentro del abstract o resumen, a esto se le llama abstract extendido, donde se identifica la estructura del modelo IMRaD. En esta subestructura no se presenta la Discusión sino la conclusión de la investigación, y en algunos abstract no se presenta la introducción sino el objetivo, por consecuente la subestructura se la determinaría de la siguiente manera: Objetivo, Métodos, Resultados y Conclusión, cuyas siglas en inglés serían "OMRC".

A través de estos modelos estructurales se determina la utilización de un modelo basado en el modelo AIMRAD, cambiando la Discusión por Conclusión. A su vez en su abstract debe identificarse la estructura OMRC, por ende el modelo estructural del artículo debe contar con las siguientes partes:

- *Abstract*
  - *Objetive* (Objetivo)
  - *Methods* (Métodos)
  - *Results* (Resultados)
  - *Conclusion* (Conclusión)
  - *Keywords* (Palabras clave)
- *Introduction* (Introducción)
- *Methods* (Metodos)
- *Results* (Resultados)
- *Discussion or Conclusion* (Discusión o Conclusión)

B. *Uso de la biblioteca NLTK*

La biblioteca NLTK es muy reconocida por su robustez para procesar información y su incremento de usuarios es exponencial dado que día a día investigadores se interesan en aprender como en aportar a elevar el nivel y eficiencia de esta biblioteca.

Como primer paso se realiza la implementación de NLTK para hallar las palabras que se repiten presentes en el artículo. Como se ilustra en la Figura 2 el código fuente para la frecuencia de palabras. Para realizar una limpieza al texto se usa el método ***RegexpTokenizer***, para eliminar los diferentes caracteres como lo son los signos de puntuación, números, etc., posteriormente se convierte el texto a minúsculas y posteriormente se tokeniza.

Las llamadas palabras vacías o "***stop words***", se caracterizan por ser palabras que no aportan al texto las cuales son los conectores que se utilizan para conformar oraciones, se las elimina con un método que lleva el mismo nombre. En este punto se utiliza un método de la biblioteca NLTK llamado ***FreqDist***, con él, se segmentan las palabras con sus respectivas frecuencias, totalizando las frases que se quiere visualizar relacionadas a su frecuencia.

```
def frecuencia_palabras(self, archivos):
    nume=len(archivos)
    for i in range(nume):
        texto = archivos[i]
        texto = str(texto)
        token = RegexpTokenizer(r'\w+')
        texto1 = texto.lower()
        tokens = token.tokenize(texto1)
        print(tokens)

        tokens_clean = [i for i in tokens if not i in en_stop]

        #Realizamos la grafica de la frecuencia de palabra
        freq = nltk.FreqDist(tokens_clean)
        pal=[word for (word, freq) in freq.most_common(10)]
        fre=[freq for (word, freq) in freq.most_common(10)]
        palabras.append(pal)
        frecuencia.append(fre)
```

Figura 2. Frecuencia de palabras

Comentario [UdW1]: No he encontrado el modelo estructural del articulo

Comentario [UdW3]: Cambiar implementación por uso, la biblioteca NLTK ya ha sido implementada

Comentario [UdW2]: Clarificar antes el significado de IMRD

C. *Uso de la biblioteca spaCy para hallar el padre sintáctico*

Es una biblioteca de uso a nivel industrial en Python que puede procesar la extracción de información en una escala mayor. Al ser más robusta que NLTK ofrece utilizar métodos con los que se predice atributos lingüísticos como dependencias nombradas, entidades nombradas y tipos de palabras, útil para predecir la categoría a la que pertenece una palabra (verbo, sustantivo, adjetivo, etc.).

Para entender cómo encontrar el padre sintáctico se debe tener en cuenta la palabra a la cual se ata una oración. Para este proceso se usa el método de spaCy llamado "***.head***", con él se obtiene todos los padres sintácticos que están presentes en la oración, frase o párrafo.

Después se procede a realizar una frecuencia de palabras para posteriormente insertarlas en un contenedor de "pandas" como se ilustra en la Figura 3.

```
nlp = spacy.load("en_core_web_lg") # se carga el modelo de vec
# matcher = Matcher(nlp.vocab) # necesitamos un vocabulario ll
nume=len(frag)
for i in range(nume):
    texto_sin=frag[i] # Añadimos el fragmento a una variable
    texto_sin=str(texto_sin) # la convertimos en string
    doc = nlp(texto_sin) # lo implementamos en un nlp de spacy
    s=[span.head.text for span in doc]
    freq = nltk.FreqDist(s)
    pal=[word for (word, freq) in freq.most_common(10)]
    fre=[freq for (word, freq) in freq.most_common(10)]
    palabras.append(pal)
    frecuencia.append(fre)
```

Figura 3. Código fuente padre sintáctico.

D. *Uso de la biblioteca spaCy para encontrar el nivel de similitud*

Aquí se usa un método para comparar dos objetos que pueden ser palabras, oraciones o trozos de un texto. El método tiene por nombre "***similarity"***, este pondera un puntaje entre 0 y 1, dando que si como resultado se obtiene aproximaciones al valor 1 entonces se deduce que su similaridad es elevada.

Esta similitud se determina con una comparación de vectores de palabras, los cuales son representaciones de significado multidimensional. Para lograr este objetivo se usa el algoritmo "***word2vec"***. Para ello es recomendable trabajar con los modelos mediano o grande porque con estos modelos se logra un mejor output con resultados que se aproximan a los deseados sin limitantes entre idiomas.

Es necesario el uso del bucle "*for*" para realizar las combinaciones de los artículos, así se puede obtener como resultado una matriz triangular. Se añade el objeto base y en otra variable el objeto con el que se va a comparar y en la penúltima línea de código se implementa el método ya mencionado para almacenar el resultado en el vector correspondiente y posteriormente guardarlo en un contenedor "DataFrame" (ver Figura 4).

```
# articulos del 1-2 al 1-6
for i in range(1,6):
    doc1 = frag[0]
    doc2 = frag[i]
    doc1_1=nlp(doc1)
    doc2_2=nlp(doc2)
    niv=doc1_1.similarity(doc2_2)
    art1_6.append(niv)
```

Figura 4. Código fuente nivel de similitud

E. *Uso del algoritmo LDA para encontrar temas*

En este proceso se puede usar la biblioteca "***Gensim"*** que contiene el modelo LDA (*LdaModel*) De la misma manera que en la frecuencia de palabras se realizó limpieza aquí existe una variación en la utilización de un algoritmo de derivación. Este algoritmo es llamado derivación de Porter y viene incorporado en la biblioteca NLTK como "***PorterStemmer()"***. Con esta derivación se encuentra la raíz (parte común) de las palabras y se pueda reducir las que sean tópicamente similares (ver Figura 5).

```
tokens_clean = [i for i in tokens if not i in en_stop]
text = [p_stemmer.stem(i) for i in tokens_clean]
textos_por.append(text)
```

Figura 5. Fragmento de código "derivación de Porter"

Es necesario crear una matriz de términos de documentos (*dictionary*), para ello se usa el método "*Dictionary()"* el cual se encuentra importando la clase "***corpora"*** de "Gensim", este método es capaz de recibir la derivación realizada anteriormente. Aquí también se crea un corpus con el método "***.doc2bow"*** que es utilizado para convertir el documento al formato de bolsa de palabras (*BoW*). Y por último se llama al método "***LdaModel"***, el cual debe enviar los siguientes parámetros:

- ***corpus***, es el flujo de vectores o la matriz dispersa representada en el formato BoW.

Comentario [UdW4]: Cambiar implementación por uso

Comentario [UdW6]: Matriz triangular

Comentario [UdW5]: Cambiar implementación por uso

- ***num_topics***, es la cantidad de temas que se extraerán del corpus.
- ***id2word***, es la asignación id de palabras a palabras *(dictionary),* se lo utiliza para determinar el tamaño del vocabulario.
- ***passes***, es el número de pasadas a través del corpus, es decir el número de veces que el algoritmo repetirá el proceso, es aquí donde se debe cuidar de que el modelo no se sobreajuste ***overfitting***.

Lo anterior se ilustra para un mejor reconocimiento del proceso en la Figura 6.

```
dictionary = corpora.Dictionary(textos_por)
corpus = [dictionary.doc2bow(text) for text in textos_por]
ldamodel = gensim.models.ldamodel.LdaModel(corpus, num_topics=3,
    id2word = dictionary, passes=80)
temas_glo=ldamodel.print_topics()
# print(temas_glo) # Vivualizacion de una lista de los temas
```

Figura 6. Fragmento de código LDA

F. Creación del Graph of Science

Se hace uso de las bibliotecas spaCy, NLTK y la biblioteca llamada "***Tesauros_NLP***", esta última es creada por Mora - Paz, Héctor Andrés y es enriquecida para la creación de un grafo.

Con spaCy como ya se ha nombrado puede realizar predicciones a nivel sintáctico es decir que podemos hallar sustantivos, verbos, adjetivos, etc., en esta ocasión se la utiliza para hallar los sustantivos presentes en el documento para posteriormente hallar su respectiva frecuencia con el método ***Counter()***, donde se almacena los cinco sustantivos más frecuentes en la variable words (ver Figura 7).

```
#Para nombres mas comunes
sustantivos=[w.text for w in nlp(texto) if w.is_stop!=True and w.is_punct!=True and w.pos_=='NOUN']
#Para Verbos mas comunes
# nombres=[w.text for w in nlp(texto) if w.is_stop!=True and w.is_punct!=True and w.pos_=='VERB']
# print(nombres)

# Colocamos los 2 sustantivos mas comunes en cada articulo y el numero de veces que es repetido
sus_freq = Counter(sustantivos)
words=[word for (word, freq) in sus_freq.most_common(5)]
```

Figura 7. Código para hallar los sustantivos más frecuentes con spaCy

Con un corpus de NLTK llamado "*wordnet*" se puede encontrar los sinónimos (*synsets*) y la definición de cada uno de los sustantivos, su resultado lo almacenamos en un "DataFrame" (ver Figura 8).

```
# Código para encontrar los synsets de los sustantivos de os articulos
lst_definition=[]
for w in words:
    for synset in wn.synsets(w):
        lst_definition.append({'key': w, 'synset': synset, 'definition':synset.definition()})
df=pd.DataFrame(lst_definition)
```

Figura 8. Código para hallar los sinónimos y su definición con wordnet

Se obtiene la matriz de los grupos de "***synsets***" con el método "***getGroups"*** para encontrar las combinaciones posibles se puede llamar al método "***combinar"***, que se encuentra en la biblioteca "***Tesauros_NLP***" en donde a la vez se llama a dos métodos más. El primero método obtiene el producto vectorial de dos vectores. El segundo método obtiene el producto vectorial de un vector y una matriz.

Para seleccionar los "*synset*" con mayor similitud (ver Figura 9), se usa el método "***MatrizTriaungularDeMetricas"*** la cual se utiliza para encontrar los niveles de similitud entre un grupo de palabras, por lo general las más importantes. Para estructurarlas se utiliza una matriz de doble entrada para calcular todas las posibles combinaciones de dos en dos utilizando una medida de distancia, estas medidas son:

- "***path_similarity***", donde su resultado es entre cero y uno, dando por hecho que entre más cerca a uno es una similitud fuerte.
- "***lowest_common_hypernyms***", la cual es para hallar el **hiperónimo** más bajo que comparten dos palabras
- ***"wup_similarity",*** esta es la similitud de Wu-Palmer, dicha similitud denota cuan similares son dos sentidos de dos palabras, según la profundidad de la taxonomía de los sentidos y de su ancestro más específico.

```
# Método para seleccionar los synset con mayor similitud
def seleccionarSynsets(verbose=False):
    groups=getGroups(words,df)
    mc=combinar(groups)
    vecmay=[]
    sumamay=0
    for vec in mc:
        md=MatrizTriangularDeMetricas(vec)
        sm=sumaMat(md)
        if(len(vecmay)==0):
            vecmay=vec
            sumamay=sm
        elif(sm>sumamay):
            vecmay=vec
            sumamay=sm
    return {"synsets":vecmay, "distancia": sumamay}
```

Figura 9. Método para hallar las distancias entre los synset

Comentario [UdW7]: Muchas gracias chicos, la cita correcta la pueden encontrar en Mora-Paz, H. A. (2019). *Comparativo de kernels sobre predicción de oferta de fuentes alternativas de energía* (Master's thesis).

Aquí está la URL https://reunir.unir.net/handle/123456789/10020

Por último, se hallan los hiperónimos de los "*synsets*" seleccionados utilizando el método hiperónimos, en donde se implementa un método de "*wordnet*" llamado "***hypernym_paths*"** (ver Figura 10). Esto es necesario para estructurar los sustantivos extraídos en los pasos anteriores, al ser palabras que pueden pertenecer a varios hiperónimos su estructura es similar a un grafo, por ello se enriquece la biblioteca "***Tesauros_NLP"*** agregando un método que permite graficar un grafo representativo (ver Figura 11).

Usando la biblioteca "*networkx"* de la documentación que se modela el grafo junto con las etiquetas, es decir, junto con los hiperónimos conectados.

```
# Método para encontrar los hyperónimos de un listado de synset
# retorna una lista de pares de valores relacionados
def hiperónimos(syntsets):
    graph=[]
    for syn in syntsets:
        for hypernyms in syn.hypernym_paths():
            n=len(hypernyms)
            for i,hypernym in enumerate(hypernyms):
                if(i<n-1):
                    n1=hypernyms[i].name().split(".")[0]
                    n2=hypernyms[i+1].name().split(".")[0]
                    graph.append((n1,n2))

    return graph
```

Figura 10. Método para hallar los hiperónimos

```
# Metodo para graficar el grafo
def graph_science(graph):
    plt.figure(figsize=(16,16))
    G=nx.Graph(graph)
    pos = nx.spring_layout(G)
    nx.draw(G,pos=pos ,with_labels=True, font_weight='bold', node_size=2000)
    for p in pos:  # raise text positions
        pos[p][1] += 500
    nx.draw_networkx_labels(G, pos)
    plt.show()
```

Figura 11. Método para generar el grafo

Según los modelos de Ingeniería Cognitiva, para modelar un grafo con "***Graph of Science***" se lo realiza con la selección de un artículo, de esta forma se ofrece una mejor claridad visual para el usuario y con ello se brinda una mejor comprensión.

V. DISCUCIÓN

Los Modelos de lenguaje GPT-2 y GPT-3 cuentan con más de 1.500 millones de parámetros para la generación de lenguaje. Open AI es una plataforma con los recursos y el capital para llevar a cabo proyectos de investigación de gran alcance a nivel mundial y para su implementación hoy en día requiere capacidades computacionales de amplio alcance como el uso de GPU y procesadores de gama alta.

Teniendo en cuenta esta referencia se ha pretendido llegar a un promedio en al que el uso de CPU aún tiene protagonismo en el tratamiento de lenguaje a nivel informático. Este referente es causal para trabajar bajo estándares que brindan al usuario el desarrollo de una herramienta escalable en futuros trabajos que aborden problemáticas similares a la planteada.

Además, es de discusión el libre acceso que la Corporación Universitaria Autónoma le pueda dar al aplicativo teniendo en cuenta que modelos educativos como el modelo prusiano en el que vivimos no es el adecuado en esta era en donde la IA toma un nuevo paradigma revolucionario que le podría dar el control en un futuro a las máquinas. Se podría afirmar que el conocimiento no es privatizable pero la economía de la sociedad podría llevar a restringir el uso de la aplicación en ambientes externos.

El desempeño abre una brecha de discusión en el que se evalúan como se aborda la similitud acercándose a valores ercanos a uno "1". Para algunos este análisis podría ser irrelevante dado el caso que los tópicos mostrarían información redundante cuando se espere un output con otro enfoque esperado antagonico de un modelado de temas.

La efectividad del algoritmo LDA se basa en un análisis lineal y las pruebas indican que el número de temas llega a un perfecto equilibrio si se logra controlar este enfoque.

VI. CONCLUSIONES

Los esfuerzos por realizar un análisis morfológico automático se basan en el estudio de la palabra del entorno de la frase más que en contemplar la palabra de forma aislada. De esta forma, todos los análisis válidos para una palabra se obtienen a partir de un diccionario léxico. Dentro del diccionario se almacena cada palabra junto a todos sus posibles análisis.

La elección del análisis correcto para la palabra dentro del contexto de la frase de entre todos los posibles asignados en el diccionario léxico para esa palabra constituye el verdadero problema del análisis morfológico de textos en inglés. A esta elección del análisis correcto

en función del contexto se denomina desambiguación morfológica. Una palabra, que es ambigua y puede pertenecer a más de una categoría gramatical, se etiqueta correctamente según el contexto de la frase analizada.

También se deben tener en cuenta las palabras vacías que en ingles son conocidas como *Stop Words*, estas palabras no aportan nada al procesamiento del artículo, por ejemplo: al realizar una frecuencia de palabras el *stop words* mostrarían una gran repetición opacando a las palabras con significado el cual es relevante para el usuario.

Para trabajos futuros la Herramienta podría evaluar el nivel de copia entre dos artículos similares. Esto se lograría modificando los valores propios de las librerías usadas y recursos programáticos más avanzados.

VII. BIBLIOGRAFÍA

[1] «WordSmith Tools». https://lexically.net/wordsmith/index.html (accedido mar. 08, 2020).

[2] «idUS - Técnicas estadísticas en minería de textos». https://idus.us.es/xmlui/handle/11441/63197 (accedido sep. 05, 2019).

[3] «v2n2a1.pdf - Ing USBMed Vol 2 No 2 Jul-Dic 2011 EL DESARROLLO DE SOFTWARE COMO INGENIERA DE SOFTWARE Scott F Schaull Columbia University NY». https://www.coursehero.com/file/29081051/v2n2a1pdf/ (accedido oct. 25, 2019).

[4] «How to Describe Alan Turing's Halting Problem to a 13 Year Old | HuffPost». https://www.huffpost.com/entry/how-to-describing-alan-turings-halting-problem-to_b_58d1ae08e4b062043ad4add7 (accedido oct. 31, 2019).

[5] «ADA TEMA 6.2 Introducción a la NP completitud - PDF». http://frasesdeamor.in/112691604-Ada-tema-6-2-introduccion-a-la-np-completitud.html (accedido oct. 25, 2019).

[6] «Generación de Lenguaje Natural: Máquinas hablando como los humanos», *DAIL Software | Automatización con Inteligencia Artificial y PLN*, abr. 11, 2019. https://www.dail.es/generacion-lenguaje-natural/ (accedido ago. 07, 2020).

[7] E. experto del sistema, «Natural Language Understanding: What is it and how is it different from NLP», *Expert System*, abr. 10, 2020. https://expertsystem.com/natural-language-understanding-different-nlp/ (accedido ago. 07, 2020).

[8] J. Vázquez Marcos, «Modelado de Tópicos para perfilado de Blogs», oct. 2017, Accedido: ago. 07, 2020. [En línea]. Disponible en: https://e-archivo.uc3m.es/handle/10016/28247.

[9] A. F. Grajales Ramírez, J. M. Molina Mejía, A. F. Grajales Ramírez, y J. M. Molina Mejía, «Current problem of anaphoric computational processing: The case of FreeLing 4.1», *Lenguaje*, vol. 47, n.º 2, pp. 537-568, dic. 2019, doi: 10.25100/lenguaje.v47i3.8356.

VIII. APENDICE

Los scripts del aplicativo como lo son la interfaz, el uso de las bibliotecas spaCy y nltk, al igual que el algoritmo LDA y el algoritmo para el graph of science, se encuentran en el siguiente repositorio https://github.com/Andrew0247/GraphOfScience.