Descubrimiento de Conocimiento Médico

Proyecto de Aprendizaje de Máquinas

Diamis Alfonso
Marié del Valle
Roxana Peña
Dennis Fiallo
Ernesto Alfonso
Rolando Sanchéz

Objetivos del proyecto

- Utilizar técnicas de machine learning y procesamiento del lenguaje natural para descubrir y extraer conocimiento médico a partir de la data de ehealthkd2021.

Data

- A corpus to support eHealth Knowledge Discovery technologies, Piad,Yoan Gutiérrez y Rafael Muñoz.
- multidominio y multilingüe.
- Todas las oraciones están relacionadas con temas de salud, mostrando gran variedad en términos de formato y etructura.
- Las oraciones esán etiquetadas, se conoce el dominio y el idioma correspondiente.
- training, tenemos un total de 1400 oraciones, de ellas hay 1200 español y 200 en inglés.
- test, tenemos un total de 100 oraciones, 75 en español y 25 en inglés.
- entidades 0.39 y relaciones 0.58

Tareas

Entre las tareas a implementar se encuentran:

1. **Named Entity Recognition (NER)**:
   Identificar y clasificar entidades en el texto. Se cuentan con 4 tipos de entidades.
   - Concept
   - Action
   - Predicate
   - Reference

Tareas

2. **Relation Extraction (RE)**:
   Identificar las relaciones y las conexiones entre las entidades en el texto. Se tienen las relaciones siguientes.
   - is-a
   - same-as
   - has-property
   - causes
   - entails
   - in-time
   - subject
   - target
   - domain
   - arg
3. **Combinación de NER y RE**.

Mediciones para NER + RE

$$Rec_{AB} = \frac{C_A + C_B + \frac{1}{2}P_A}{C_A + I_A + C_B + P_A + M_A + M_B}$$

$$Prec_{AB} = \frac{C_A + C_B + \frac{1}{2}P_A}{C_A + I_A + C_B + P_A + S_A + S_B}$$

$$F_{1AB} = 2 \cdot \frac{Prec_{AB} \cdot Rec_{AB}}{Prec_{AB} + Rec_{AB}}$$

Mediciones para NER

$$Rec_A = \frac{C_A + \frac{1}{2}P_A}{C_A + I_A + P_A + M_A}$$

$$Prec_A = \frac{C_A + \frac{1}{2}P_A}{C_A + I_A + P_A + S_A}$$

$$F_{1A} = 2 \cdot \frac{Prec_A \cdot Rec_A}{Prec_A + Rec_A}$$

Mediciones para RE

$$Rec_B = \frac{C_B}{C_B + M_B}$$

$$Prec_B = \frac{C_B}{C_B + S_B}$$

$$F_{1B} = 2 \cdot \frac{Prec_B \cdot Rec_B}{Prec_B + Rec_B}$$

LSTM (Long Short-Term Memory)

Definición

Es un tipo de red neuronal recurrente (RNN) que se utiliza para procesar y analizar secuencias de datos, como texto, audio o series de tiempo. A diferencia de las RNN tradicionales, que pueden tener dificultades para capturar relaciones a largo plazo en los datos, las LSTM están diseñadas específicamente para resolver este problema mediante el uso de celdas de memoria.

BERT (Bidirectional Encoder Representations from Transformers)

Definición

Es un modelo de lenguaje basado en la arquitectura Transformer desarrollado por Google. A diferencia de los modelos de lenguaje tradicionales, que se entrenan de manera unidireccional, BERT se entrena de manera bidireccional, lo que le permite capturar el contexto de las palabras en una oración de manera más efectiva.

T5 (Text-To-Text Transfer Transformer)

Definición

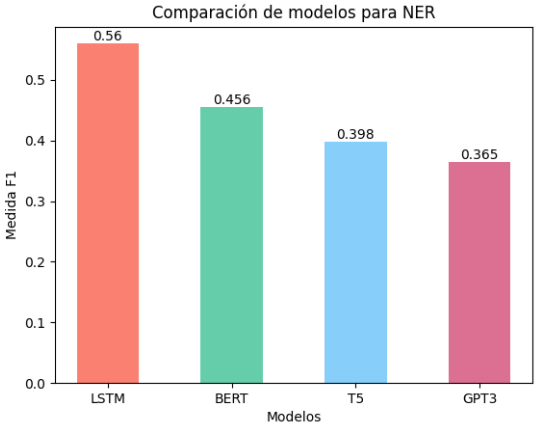
Es un modelo de lenguaje basado en la arquitectura Transformer y diseñado específicamente para la transferencia de texto a texto, lo que significa que puede abordar una amplia gama de tareas de procesamiento del lenguaje natural (NLP) mediante la formulación de la entrada y la salida en un formato de texto a texto.

GPT-3 (Generative Pre-trained Transformer 3)

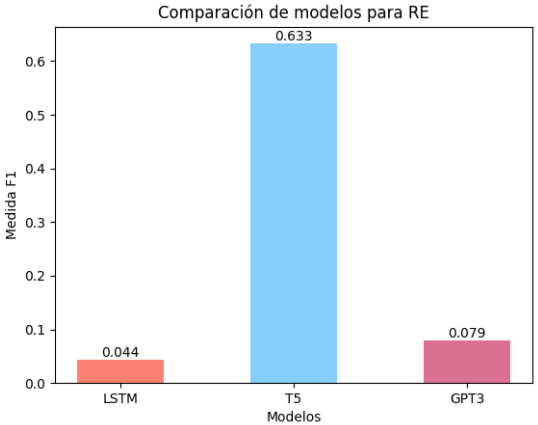
Definición

Es un modelo de lenguaje desarrollado por OpenAI. Es la tercera versión de la serie GPT y se basa en la arquitectura Transformer. Puede generar texto coherente y contextualmente relevante en respuesta a una entrada de texto.

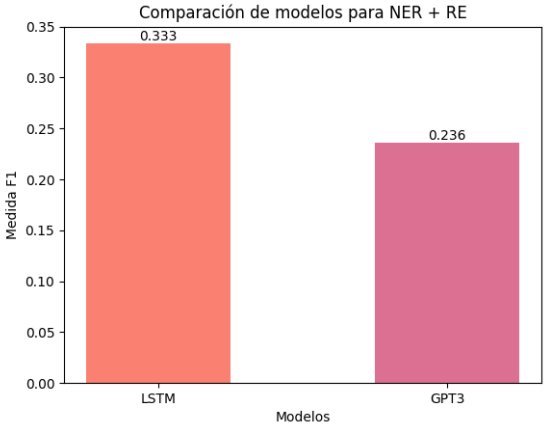
NER


Comparación de modelos para NER
Medida F1
Modelos
0.56
0.456
0.398
0.365
LSTM
BERT
T5
GPT3
0.0
0.1
0.2
0.3
0.4
0.5

RE


Comparación de modelos para RE
Medida F1
Modelos
LSTM
T5
GPT3
0.044
0.633
0.079
0.0
0.1
0.2
0.3
0.4
0.5
0.6

Combinación NER y RE



Ontologías

Una vez obtenido todo el conocimiento de los documentos, se guardan los datos en una base de datos orientada a grafos. Para ello se utiliza **Neo4j**, el cual permite acceder a los datos de diversas formas y usando distintos lenguajes de consulta. En nuestro proyecto se utiliza **Cypher**, un lenguaje que permite consultar y manipular grafos.

Ontologías

La comunicación con la base de datos se realiza en el siguiente método:

```
def create_database(self):
    relations, keyphrases = OntologyUtils.load_result()
    with self.driver.session(database="neo4j") as session:
        # Write transactions allow the driver to handle retries and transient errors

        for key,value in keyphrases.items():
            result = session.execute_write(
                self.create_entity, value, key
                )

        for key,value in relations.items():
            result = session.execute_write(
                self.create_relation, key[0], key[1], key[2], key[3], value
            )
```

Ontologías

La generación de los nodos como entidades la realizamos con las siguientes consulta:

```
@staticmethod
def create_entity(tx, value, key):
        if value == "Concept":
            query =(
                "CREATE (w:Concept { text: $key }) "
                "RETURN w"
            )
        elif value == "Action":
            query =(
                "CREATE (w:Action { text: $key }) "
                "RETURN w"
            )
        elif value == "Predicate":
            query =(
                "CREATE (w:Predicate { text: $key }) "
                "RETURN w"
            )
        elif value == "Reference":
            query =(
                "CREATE (w:Reference { text: $key }) "
                "RETURN w"
            )
        result = tx.run(query, value=value, key=key)
```

Ontologías

Para la creación de las relaciones entre los nodos utilizamos la consulta:

```
@staticmethod
def create_relation(tx, key0, key1, key2, key3, value):
        query = (
        "MATCH (p {text: $key0}), (r {text: $key2})"
        "CREATE (p)-[: is_related {relation: $value}]->(r) "
        )

        result = tx.run(query, key0=key0, key1=key1, key2=key2, key3=key3, value=value)
```

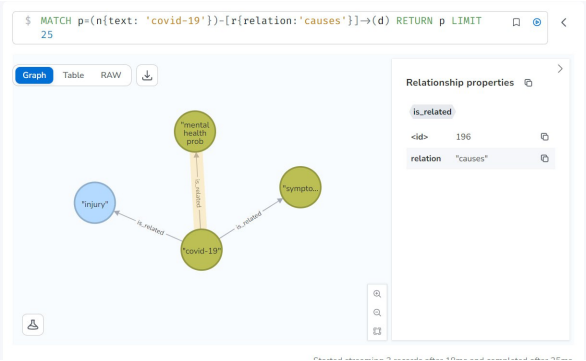
Ontologías

Este método realiza una consulta para conocer las causas de alguna enfermedad de interés:

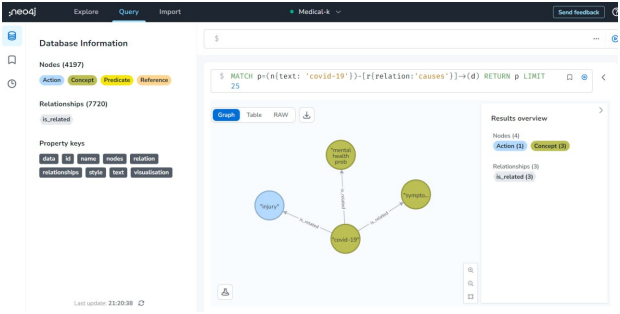
```
@staticmethod
def makeQueryd(tx, disease):
    query =(
        "MATCH p=(n{text: $disease})-[r{relation:'causes'}]->(d) RETURN d LIMIT 25"
    )
    result = tx.run(query, disease=disease)
    listNode = [node for node in result]
    return listNode
```

Ontologías

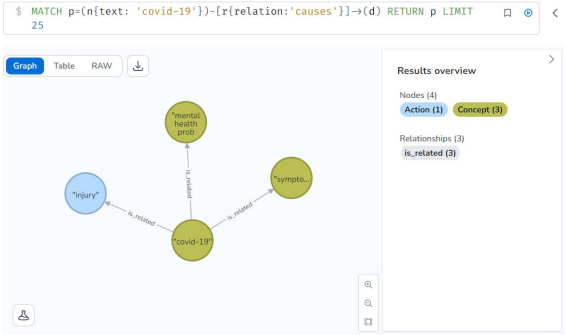
En esta imagen podemos ver los resultado obtenidos al realizar la consulta para conocer las causas del covid-19:



Ontologías


neo4j
Explore
Query
Import
Medical-k
Send feedback
Database Information
Nodes (4197)
Action
Concept
Predicate
Reference
Relationships (7720)
is_related
Property keys
data
id
name
nodes
relation
relationships
style
text
visualisation
Last update: 21:20:38
$
$ MATCH p=(n{text: 'covid-19'})-[r{relation:'causes'}]->(d) RETURN p LIMIT 25
Graph
Table
RAW
"mental health prob
"injury"
"sympto
"covid-19"
is_related
Results overview
Nodes (4)
Action (1)
Concept (3)
Relationships (3)
is_related (3)

Ontologías


$ MATCH p=(n{text: 'covid-19'})-[r{relation:'causes'}]→(d) RETURN p LIMIT 25
Graph
Table
RAW
"mental health prob
"injury"
"sympto...
"covid-19"
is_related
is_related
is_related
Results overview
Nodes (4)
Action (1)
Concept (3)
Relationships (3)
is_related (3)

Muchas Gracias