8/12/2022

**Calcular la similitud semántica entre ideas.**

Usaremos algoritmos no supervisados: al no tener datos etiquetados, buscamos que los algoritmos encuentren grupos o clústers de ideas (tópicos). Por ejemplo, ideas políticas de izquierda y de derecha, o ideas liberales y conservadoras. Esta tarea es conocida como “**Topic Modeling**”.

Luego, podremos comparar la similitud semántica entre: ideas con ideas, ideas con tópicos, y tópicos con tópicos. Esta tarea es conocida como “**Semantic Similarity**”.

Probaremos con algoritmos de Deep Learning, por un lado, y de Machine Learning clásico/no Deep Learning por el otro.

Todos los algoritmos generan un vector para cada tópico, para cada documento y para cada palabra. Podemos calcular la similitud coseno entre los distintos vectores como una medida de similitud semántica. Podemos hacer otras cosas, como reducir su dimensionalidad y graficarlos.

**Algoritmos de Machine Learning:**

**1) TF-IDF**

* Es el algoritmo más sencillo.
* Genera una matriz con todas las palabras únicas en el dataset en las filas, y todos los documentos (cada opinión de cada participante) en las columnas. Luego, rellena los casilleros con la cantidad de veces que cada palabra aparece en cada documento. Finalmente, divide cada casillero por la cantidad de documentos que tienen esa palabra. En otras palabras, calcula que tan representativa es cada palabra para cada documento.
* Cada documento resulta en un vector de V dimensiones (V = cantidad de palabras únicas en todo el dataset).
* Cada palabra resulta en un vector de N dimensiones (N = cantidad de documentos).
* Las similitudes semánticas pueden compararse con similitud coseno entre los distintos vectores.

Ventajas:

* Alta interpretabilidad: el vector de un documento se compone de los pesos asignados a cada palabra. El peso de cada palabra refleja que tan importante es dicha palabra para diferenciar semánticamente ese documento del resto.

Desventajas:

* Solo sirve para evaluar Semantic Similarity (es decir, no encuentra tópicos por sí solo). Sin embargo, pude combinarse con un algoritmo de clústering (como K-means) y de esta manera encontrar tópicos.
* Es un algoritmo de “bolsa de palabras”: no importa el orden de las palabras en el texto, solo las palabras en sí mismas. Esto significa que “Argentina es corrupta” y “Argentina no es corrupta” serían evaluadas como muy similares y posiblemente asignadas al mismo clúster.

**2) Latent Semantic Analysis (LSA).**

* Es el algoritmo más utilizado para Semantic Similarity.
* Genera una matriz con todas las palabras únicas en las filas, y todos los documentos en las columnas (llenando las celdas con la cantidad de veces que la palabra se encuentra en el documento). Luego, utiliza el algoritmo llamado Singular Value Decomposition para reducir la dimensionalidad de la matriz y crear vectores para cada tópico, para cada documento y para cada palabra.
* El número de tópicos es definido por el propio algoritmo automáticamente (si queremos, luego podemos quedarnos con los top K más relevantes).

Ventajas:

* Fácil y rápido de entrenar con pocos datos.
* Alta interpretabilidad (el vector de documento y el vector de palabra, se componen de pesos para cada tópico). El algoritmo puede entenderse por completo.

Desventajas:

* Es un algoritmo de “bolsa de palabras”.

**3) Latent Dirichlet Allocation (LDA).**

* Es un modelo probabilístico generativo: cada documento y cada palabra tiene un vector con la probabilidad de pertenecer a cada tópico.
* El número de tópicos es un hiperparametro a definir.
* Las similitudes semánticas pueden compararse con distancia coseno entre los distintos vectores.

Ventajas:

* Fácil y rápido de entrenar con pocos datos.
* Alta interpretabilidad (el vector de documento y el vector de palabra, se componen de pesos para cada tópico). El algoritmo puede entenderse por completo. Sin embargo, es más complejo que los anteriores dos.
* Menos propenso a over-fitting comparado a LSA.

Desventajas:

* Es un algoritmo de “bolsa de palabras”.
* Requiere más tiempo, más optimización de hiperparametros, más poder de cómputo.

**Algoritmos de Deep Learning:**

**1) LSTM**

* Es un tipo de red neuronal recurrente.
* Permite realizar topic modeling y semantic similarity.

Ventajas:

* NO es un algoritmo de bolsas de palabras. El contexto en el que aparecen las palabras es tenido en cuenta. Esto significa que “Argentina es corrupta” y “Argentina no es corrupta” tendrán significados opuestos.
* Comparado al resto de los algoritmos de Deep Learning, podría funcionar mejor con pocos datos.

Desventajas:

* Muchos hiperparametros.
* Demanda mucho poder de cómputo (predicciones más lentas).
* Baja interpretabilidad. El algoritmo no puede entenderse por completo.

**1) Transformers**

* Estado del arte en NLP con Deep Learning.
* Es un algoritmo basado en mecanismos de atención.
* Permite realizar topic modeling y semantic similarity.
* Es todo un mundo de modelos: BERT, BERTopic, TinyBert, etc.

Ventajas:

* NO es un algoritmo de bolsas de palabras. El contexto en el que aparecen las palabras es tenido en cuenta.
* Probablemente obtenga el mejor rendimiento.

Desventajas:

* Muchos hiperparametros
* Demanda aún más poder de cómputo (predicciones más lentas).
* Baja interpretabilidad. El algoritmo no puede entenderse por completo.

**Evaluación de los modelos:**

Para evaluar los modelos, veremos qué tan coherentes y separables entre sí son los tópicos. Tenemos al menos tres formas de evaluarlo:

1. Inspección visual: ver las palabras y documentos con más peso en cada tópico, y evaluar si es coherente. También podemos utilizar un algoritmo de reducción de dimensiones para graficar las palabras de cada tópico, y luego ver que tan coherentes y separables son los grupos.
2. Similitud de vectores de palabras: obtenemos un vector para cada palabra. Luego, para cada tópico por separado, calculamos la similitud coseno entre cada par posible de vectores y obtenemos la similitud promedio. Cuanta más similitud interna para cada uno de los tópicos, mejor el modelo.
3. Similitud de vectores de documento: lo mismo, pero con los vectores de los documentos.

**Conclusiones:**

* Estos métodos no son excluyentes, deberíamos probar todos.
* Existe un trade-off (una relacion inversamente proporcional) entre la eficacia de un algoritmo y su interpretabilidad y tiempo de cómputo (entrenamiento y predicción).
* Podemos usar los algoritmos simples para interpretar, y los complejos para realmente asignar tópicos.

**Algunas otras ideas:**

* Al tener un vector para cada documento, podemos evaluar la similitud coseno entre cada par posible de vectores. El documento con mayor similitud promedio, es el vector más representativo del tópico.
* Separar todas las ideas con Sentiment Analysis, y luego ver las palabras clave mas asociadas a cada sentimiento.
* Realizar Named Entity Recognition: identificar las personas, lugares, eventos y fechas más mencionados.
* Utilizar técnicas de sumarización de texto para resumir cada tópico en una o dos frases, y ver más fácilmente las ideas mas representativas.
* Técnicas de Análisis de Discurso para analizar la forma en la que se construyen los argumentos.