6 de junio de 2023

Fine-tuning de GPT-2 para diagnósticos

EQUIPO:
David Salvador Preciado Márquez
Britny Brito Juárez
Juan José Arroyo Rivera
José Manuel Pedro Méndez
Kevin Ariel Merino Peña

RECONOCIMIENTO DE PATRONES
Y APRENDIZAJE AUTOMATIZADO

Introducción

Los modelos grandes de lenguaje (LLM) han resaltado en los últimos años por su gran desempeño en una variedad de tareas. GPT-2 es uno de estos modelos, fue creado por OpenAI y entrenado de manera no supervisada sobre una gran cantidad de texto obtenido en internet. Este modelo generalista se puede adecuar realizando *fine-tuning*, que consiste en entrenar nuevamente el modelo ya preentrenado pero ahora sobre un conjunto de datos personalizado a una tarea específica. Este modelo puede considerar una mayor cantidad de posibilidades gracias a su conjunto de entrenamiento original, con lo cual no se encuentra limitado a los casos que aparecen en un conjunto de datos específico.

Objetivo

El propósito de este proyecto es hacer *fine-tuning* de GPT-2 usando un conjunto de datos de diagnósticos médicos. Con esto buscamos obtener un modelo con la capacidad de generar diagnósticos precisos tomando como entrada una descripción de síntomas. Compararemos este modelo con otros modelos obtenidos con una variedad de métodos distintos, para así determinar la manera óptima de aprovechar el conjunto de datos y obtener el mejor desempeño posible.

Dataset

El conjunto de datos consiste en frases en inglés que son pares de descripciones de síntomas junto con su diagnóstico correspondiente. Estos datos fueron obtenidos de *Hugging Face* [6], y a su vez estos vienen de un *web scraping* de la página web de *Mayo Clinic* [5].

Método

En este proyecto se compara el desempeño de modelos distintos entrenados con el mismo conjunto de datos, a continuación se ofrece una descripción de los mismos.

GPT-2

Este modelo sigue la arquitectura conocida como transformador, que se destaca por su uso del mecanismo de autoatención con el cual asigna relevancia a cada parte de la entrada en comparación con todas las demás. El modelo procesa el texto de entrada mediante dos pasos clave, el *tokenizer* y el *embedding*.

El *tokenizer* se encarga de separar la oración de entrada en componentes semánticos significativos. Dado que estamos utilizando un modelo preentrenado y este espera cierto formato para las entradas, usamos la implementación de *tokenizer* para GPT-2 que puede ser revisada en su repositorio original [7]. En resumen, emplea la siguiente serie de pasos [8]:

1. Se preprocesa el texto para manejar caracteres especiales. En general usamos *Byte Pair Encoding* (BPE) que consiste en representar palabras como secuencias de subpalabras.
2. Se realiza el proceso sobre las palabras usando como base el conjunto del punto anterior. Por ejemplo, para ”unhappiness”: ”un”, ”hap”, y ”pines”
3. A los caracteres especiales del primer punto se les agrega su propio *token*, y a cada uno de los *tokens* creados se les asigna un identificador.

Después de aplicar el *tokenizer* a las oraciones se procede al *embedding*, que consiste en transformar las oraciones procesadas en vectores numéricos. La arquitectura de transformador ya incluye un componente que se entrena con

retropropagación para determinar *embeddings* de tal forma que *tokens* similares semánticamente tengan una representación vectorial similar. A esto se le añade una codificación posicional, y se alimenta hacia adelante al siguiente componente de la red.

Utilizamos un modelo GPT-2 preeentrenado con paradigma causal para predecir el siguiente token dada una secuencia de *tokens*, aprovechando la biblioteca `Transformers` de *Hugging Face* [3]. Actualizamos los pesos mediante retropropagación usando el algoritmo *AdamW* [4], con una tasa de aprendizaje de 0.00002 y un decaimiento de 0.01.

Para el entrenamiento usamos como entrada la concatenación las descripciones de síntomas con preguntas al estilo de *"What diagnosis would you give to a patient with the previous symptoms?"*. Esto nos permite usar un entrenamiento no supervisado, las salidas esperadas consisten en lo anterior con el diagnóstico correcto concatenado.

LSTM

Este modelo sigue la arquitectura de una Red Neuronal Recurrente de Long Short-Term Memory (LSTM) bidireccional, que destaca por su capacidad de entender las dependencias de largo plazo en los datos de secuencia, considerando tanto el contexto pasado como el futuro de cada punto en la secuencia de texto.

El modelo procesa el texto de entrada mediante tres pasos clave: el *tokenizer*, el *padding* y el *embedding*[9].

La implementación de *tokenizer* en este caso es proporcionada por Keras y es específica para el conjunto de datos de entrenamiento. Posteriormente se realiza el *padding* para asegurar que todas las secuencias de palabras tengan la misma longitud. Para este modelo, la longitud máxima se establece en base al texto más largo en el conjunto de datos. Después se lleva a cabo el *embedding*, la capa de *embedding* se entrena junto con el modelo para aprender representaciones vectoriales semánticamente significativas de las palabras.

El núcleo del modelo es la capa LSTM bidireccional. Estas también toman en cuenta el contexto futuro. Esto se logra alimentando la secuencia de entrada en dos LSTMs diferentes, una que procesa la secuencia de entrada en el orden original y otra que la procesa en orden inverso. Las salidas de ambas LSTMs se combinan antes de pasar al siguiente nivel de la red.

Por último tenemos una capa densa con activación *softmax* que clasifica la salida de la LSTM bidireccional en una de las categorías de etiquetas, que representan diagnósticos médicos en este caso. Esta función se asegura de que las salidas del modelo sean probabilidades que suman 1, y la categoría con la mayor probabilidad se elige como la predicción del modelo. Para el entrenamiento se utiliza el optimizador *Adam* y la función de pérdida de entropía cruzada categórica para etiquetas codificadas como números enteros. La métrica de evaluación es la precisión. Los pesos del modelo se actualizan mediante retropropagación a lo largo de múltiples épocas, establecido en 32.

Los datos de entrada para el entrenamiento son la descripción de los síntomas, y las salidas esperadas son los diagnósticos correspondientes, lo que permite un entrenamiento supervisado.

Naive Bayes Classifier

Se utilizó un Tf-IDF vectorizer [2]para procesar el texto como un vector bajo el modelo de bag-of-words Por lo que se hizo un preprocesamiento del texto, quitando signos de puntuación y stopwords para evitar taggear palabras repetidas en el idioma como significativas Y se entrenó el MultinomialNB de Keras con un parámetro alpha= 1.3 [1]

Si fuera un dataset con múltiples filas por cada label relacionada a una enfermedad, se podría hacer clasificación, pero dada la naturaleza de los datos en este caso no fue posible

Resultados

Para evaluar y comparar los modelos utilizamos la métrica de perplejidad, definida para modelos de lenguaje como:

$$PP(W) = \sqrt[N]{\frac{1}{P(w_1, w_2, \ldots, w_N)}}$$

Donde W es el conjunto de prueba, que es la secuencia de palabras de todas todas las oraciones concatenadas incluyendo tokens de inicio y final de oración (por ejemplo, W =(<SOS>, Esta, es, una, oración, <EOS>, <SOS>, Esta, es, otra, oración, <EOS>) y N es la cantidad de tokens en todo el conjunto de prueba. Una menor perplejidad indica un mejor modelo. Además, proveemos ejemplos de inferencia con el modelo entrenado.

Fine-tuned GPT-2

Después del entrenamiento obtuvimos una perplejidad de 7.70 (como referencia, GPT-2 sin *fine-tuning* obtuvo una perplejidad de 18.26). Haciendo inferencia sobre nuevas frases que no están presentes en el conjunto de datos obtuvimos salidas coherentes del siguiente estilo:

```
Prompt: The patient has fever, throat pain and headhache.
        What would be his diagnosis?
Output: Influenza
```

También obtuvimos salidas no coherentes, que de hecho conformaron la mayoría de las salidas del modelo.

```
Prompt: The patient has fever, throat pain and headhache.
        What would be his diagnosis?
Output: hairdresser
```

LSTM

Desafortunadamente después del entrenamiento obtuvimos una precisión del 0.1217, es decir, solo alrededor del 0.1217 % de las predicciones realizadas por el modelo son correctas. Esto es bastante bajo y esperado, ya que nuestro modelo puede tener problemas de aprendizaje por el tamaño de nuestro conjunto de datos y la cualidad de los datos. También podemos observar que nuestra perplejidad es bastante alta a comparación del *fine-tuning* de GPT-2.

Haciendo inferencia sobre frases que están presentes en el conjunto de datos para el test obtuvimos salidas del siguiente estilo:

```
Prompt: The patient has fever, throat pain and headhache.
        What would be his diagnosis?
Output: dermatomyositis

Prompt: The patient has a swollen mole that is itchy and reddish in color.
        What would be his diagnosis?
Output: vaginitis
```

Conclusiones

En todos los modelos encontramos que el conjunto de datos utilizado resultó ser muy pequeño como para la complejidad que involucra el problema. A pesar de esto, con el *fine-tuning* de GPT-2 logramos mejorar su desempeño respecto a su versión normal y además pudimos obtener algunas inferencias sencillas. No podemos decir lo mismo de los otros modelos, que en general tuvieron un desempeño pobre. Esto nos indica que es importante contar con el contexto adicional y la generalización que provee un modelo grande de lenguaje, pues permite sacar mayor provecho de conjuntos pequeños de datos.

Referencias

[1] SkLearn API docs. *Naive Bayes*. 2023. URL: https://scikit-learn.org/stable/modules/naive_bayes.html.

[2] SkLearn Tf-IDF docs. *TF-IDF Vectori*. 2023. URL: https://scikit-learn.org/stable/modules/generated/sklearn.feature_extraction.text.TfidfVectorizer.html.

[3] Hugging Face. *OpenAI GPT2*. URL: https://huggingface.co/docs/transformers/model_doc/gpt2.

[4] Ilya Loshchilov y Frank Hutter. *Decoupled Weight Decay Regularization*. 2019. arXiv: 1711.05101 [cs.LG].

[5] *Mayo Clinic*. 24 de abr. de 2023. URL: https://www.mayoclinic.org/.

[6] *Mayo Clinic Symptoms and Diseases*. URL: https://huggingface.co/datasets/celikmus/mayo_clinic_symptoms_and_diseases_v1.

[7] OpenAI. *GPT-2 Tokenizer*. URL: https://github.com/openai/gpt-2/blob/master/src/encoder.py.

[8] Luke Salamone. *How does GPT-2 Tokenize Text?* 18 de jun. de 2021. URL: https://lukesalamone.github.io/posts/gpt2-tokenization/.

[9] Enes Zvornicanin. *Differences Between Bidirectional and Unidirectional LSTM*. URL: https://www.baeldung.com/cs/bidirectional-vs-unidirectional-lstm.