CLASIFICACIÓN MULTIETIQUETA DE DIFERENCIAS EN VERSIONES DE SLA

MACHINE LEARNING ENGIEERING

MÁSTER EN INGENIERÍA DEL SOFTWARE - CLOUD, DATOS Y GESTIÓN TI

JUAN CARLOS CORTÉS MUÑOZ
MARÍA ELENA MOLINO PEÑA

REPOSITORIO EN GITHUB

Contenido

Temática y objetivos 2
Datos 3
Detección de diferencias 4
Técnicas y herramientas 4
  Zero-Shot con BART 5
  Ajuste del modelo BERT 5
  Few-Shot 6
Publicación de modelos en Hugging Face 8
Resultados 8
Conclusiones 9
Bibliografía 9

Temática y objetivos

Este proyecto tiene como objetivo investigar los diversos aspectos de aplicación del procesamiento de lenguaje natural a través de la implementación de diferentes modelos Transformers para clasificar oraciones en múltiples categorías. La finalidad es crear una base teórica y experimental sólida sobre la biblioteca Hugging Face que pueda ser utilizada como punto de partida en futuros proyectos de investigación relacionados.

El problema de estudio que se plantea esta relacionado con la comparación entre versiones de Acuerdos a Nivel de Servicio. A grandes rasgos, un Acuerdo a Nivel de Servicio es un documento contractual entre un cliente y un proveedor donde se establecen las expectativas de rendimiento de los servicios cubiertos [1].

Por lo tanto, un SLA está compuesto por un conjunto de características medibles (métricas) para las que se define un nivel o rango de objetivos (denominados objetivos a nivel de servicio), esto permite medir el rendimiento en diferentes intervalos de tiempo. En ocasiones, el proveedor no cumple con los objetivos definidos, para ello, suelen estar definidas las compensaciones (crédito por ejemplo monetario).

De forma general, el cliente debe solicitar los créditos de servicio según se describe en el proceso de reclamación del SLA. Es esencial tener en cuenta que el acuerdo debe contener las excepciones en las que el proveedor no se hace responsable de cumplir con los objetivos de nivel de servicio. Por último, suelen indicar las definiciones de los términos específicos del dominio del problema [2].

Las versiones de los Acuerdos a Nivel de Servicio suelen cambiar frecuencia en el tiempo. Por ello, detectar de forma automática las diferencias y clasificarlas en diferentes categorías puede ser de interés para los clientes de los servicios. Dada la definición anterior, las categorías que se plantean son las siguientes:

- Service, hace referencia a los servicios en la nube ofrecidos por el proveedor incluidos en el SLA.
- Metric, define cómo debe medirse una propiedad de servicio y en qué contexto. En otras palabras, es una norma de medición que especifica las condiciones (por ejemplo, el intervalo de tiempo), las reglas (por ejemplo, la fórmula de cálculo) y la interpretación de los resultados (por ejemplo, SLO) con respecto a la medición de una propiedad de servicio.

- Objective, valor específico o rango de valores que el proveedor se compromete a entregar sobre una característica específica medible del servicio en nube.
- Remedy, compensación disponible para el cliente en caso de que el proveedor no cumpla un objetivo de nivel de servicio especificado. Para ello, suelen definirse diferentes importes de compensación que se asocian a los objetivos de nivel de servicio. Estas pueden adoptar diferentes formas, como reembolsos de cargos, servicios gratuitos u otros.
- Claims process, método que debe seguir el cliente para reclamar créditos en caso de que el proveedor incumpla el acuerdo.
- Exception, describe las circunstancias en las que no se aplica la SLO. Ejemplos de excepciones: cortes programados, catástrofes naturales, etc.
- Definition, son términos exclusivos del proveedor o especialmente importantes para comprender el acuerdo.

Debido a la falta de datos estructurados o procesados en el dominio a tratar, consideramos buena idea generar un dataset reducido que permita entrenamiento al menos Few-Shot. Para ello, se han extraído las oraciones de diversos acuerdos y clasificado de forma manual en las categorías anteriormente nombradas.

Las fases que se ejecutan durante el proceso de desarrollo de este proyecto son las siguientes:

1. Extracción de datos, clasificación y generación del dataset. En los documentos anexos "train_dataset.pdf" y "validation_dataset.pdf" se aporta más detalles de la creación del conjunto de datos.
2. Análisis y preprocesado. Antes de realizar el entrenamiento se observa si la frecuencia de ejemplos en cada clasificación es similar.
3. Entrenamiento de modelos. Aplicación de Transformers desde la librería Hugging Face. Los pasos son preparación de los datos, entrenamiento, evaluación e inferencia.
4. Subir modelos entrenados a Hugging Face. La herramienta plantea la posibilidad de publicar modelos entrenados y conjunto de datos generados.

Datos

Las oraciones contenidas en el set de datos se han adquirido de cinco acuerdos de nivel de servicio de distintos proveedores y están disponibles en la web. Entre ellos Amazon[1] , Google[2],

[1] https://aws.amazon.com/compute/sla/
[2] https://cloud.google.com/appengine/sla

Oracle[3] y Twilio[4] empleados en el conjunto de entrenamiento y Microsoft[5] para el conjunto de validación.

Para generación del dataset vacío se ha creado un script que divide el texto en oraciones gracias al paquete "spacy" (ver generate_dataset.ipynb). Posteriormente, se realiza un análisis de las oraciones con la inteligencia natural de los autores. La información que contienen el dataset final es la que se muestra en la Ilustración 1.

	text	service	metric	objective	remedy	claim	exception	definition	obligation	right	neither
0	Last Updated: May 25, 2022 This Amazon Compute...	1	1	0	0	0	0	0	0	0	1
1	In the event of a conflict between the terms o...	1	0	0	0	0	0	1	0	0	1
2	Capitalized terms used herein but not defined ...	1	0	0	0	0	0	0	0	0	1
3	*For purposes of this SLA, Amazon EC2 includes...	1	1	0	0	0	0	0	0	0	1
4	SLAs AWS makes two SLA commitments for Amazon ...	1	1	0	0	0	0	0	0	0	1

Ilustración 1: Extracto del conjunto de datos

Detección de diferencias

En cuanto a la búsqueda de diferencias entre dos versiones de un mismo SLA, se genera un Script que emplea algunos paquetes para detectar la similitud entre oraciones y las distinciones semánticas (ver get_differences.ipynb). De forma general el funcionamiento es que dado dos textos, extrae las diferencias y se genera un Dataset para su posterior inferencia en los modelos entrenados.

Técnicas y herramientas

En esta sección se realiza un estudio de la biblioteca Hugging Face y las posibilidades que dispone para resolver el problema que se plantea. La herramienta de procesamiento de lenguaje natural incluye multitud de modelos Transformers pre-entrenados y enfocados en resolver gran variedad de tareas, entre ellas la técnica más adecuada para esta cuestión es la clasificación de oraciones con múltiples etiquetas [3].

Además de poder emplear modelos pre-entrenados para tareas específicas, se puede realizar un entrenamiento de estos en base a un corpus de datos del dominio del problema. Lo que permite una mayor flexibilidad y ajuste a las necesidades de cada proyecto.

En este contexto, se seleccionan varias formas de entrenamiento y modelos, comparando los resultados obtenidos, abordando un mayor número de alternativas teniendo en cuenta el

[3] https://www.oracle.com/assets/paas-iaas-pub-cld-srvs-pillar-4021422.pdf , pág. 35
[4] https://www.twilio.com/legal/service-level-agreement
[5] https://www.azure.cn/en-us/support/sla/kubernetes-service/

escaso volumen de los datos que se disponen. Por un lado, se lleva a cabo una prueba de concepto aplicando la técnica de entrenamiento de tiro de cero (zero shot) con el modelo “facebook/bart-large-mnli” disponible en Hugging Face, obteniendo unos resultados completamente aleatorios. En segundo lugar, se aplica fine tunning estándar al modelo BERT con customización de los argumentos del entrenamiento. Por último, se emplea el método de aprendizaje más eficiente con una cantidad reducida de datos, entrenamiento de tiro de pocos (few shot), con los modelos “paraphrase-MiniLM-L3-v2”, “all-MiniLM-L6-v2” y “paraphrase_mpnet_base_v2”.

A continuación, se especifican los modelos empleados y se detalla su aplicación en este proyecto.

Zero-Shot con BART

BART (Bidirectional and Auto-Regressive Transformer) es un modelo de lenguaje de aprendizaje profundo desarrollado por Facebook. Es una variante mejorada de la arquitectura Transformer (secuencia a secuencia), que ha sido ampliamente utilizada en el procesamiento de lenguaje natural. La versión "facebook/bart-large-mnli" disponible en Hugging Face ha sido entrenada en una amplia variedad de tareas, entre ellas las de tiro de cero.

La finalidad de ejecutar este modelo en el problema propuesto es abarcar al mayor número de técnicas posibles teniendo en cuenta las limitaciones del dataset que se dispone. El proceso que se ha empleado para aplicar este modelo comienza con la carga del modelo desde la librería. Posteriormente obtenemos la lista de clasificaciones reales y la predicción del modelo. Finalmente, se calcula la medida de bondad Accuracy. (ver bart_large_classification.ipynb).

Ajuste del modelo BERT

BERT (Bidirectional Encoder Representations from Transformers) es un modelo de lenguaje de aprendizaje profundo desarrollado por Google. Se trata de un modelo que emplea la tecnología Transformer. Ha sido entrenado para predecir la próxima palabra en una secuencia de texto utilizando una técnica de codificación bidireccional. Presenta resultados efectivos en tareas de clasificación multietiquetas como el que se plantea en este proyecto [5].

El objetivo del trabajo es entrenar el modelo "bert-base-uncased" para realizar la clasificación de oraciones en relación a Acuerdos a Nivel de Servicio. El proceso comienza con la selección y procesamiento de los datos de entrenamiento. En este contexto, el primer paso es la generación del Dataset como se explica con anterioridad.

Tras esto, es necesario tokenizar los datos para poder entrenar el modelo. Es decir, las oraciones se dividen en secuencia de tokens obteniendo como salida los inputs id, token types ids y attention mask, a lo que posteriormente se añade el conjunto de etiquetas. La creación del TensorDataset a partir de los tensores de entrada y las etiquetas para realizar el entrenamiento en lotes es primordial.

A continuación, se importa el modelo indicando el tipo de problema a tratar y se definen los argumentos de entrada a la clase Trainer disponible en Hugging Face. Entre ello, el número de épocas que se va a ejecutar el modelo, la cuota de entrenamiento, el tamaño del lote y la métrica más adecuada para la selección del mejor modelo entrenado.

Posteriormente, se realiza el entrenamiento del modelo empleando como función de perdida BCEWithLogitsLoss, ya que según se ha estudiado en los ejemplos propuestos en Hugging Face se plantea que la función de pérdida de modelo BERT está diseñada para evaluar todas las probabilidades de las categorías individualmente en lugar de en comparación con otras categorías. Por ello, se emplea BCE en lugar de Entropía Cruzada a la hora de definir la pérdida.

Tras el entrenamiento, se evalúa el rendimiento del modelo en la tarea de clasificación. Para ello, la métrica principal en la que se centra el sistema es Accuracy, puesto que la distribución de las clases es equilibrada. Aunque también se proporcionan otras métricas como son F1-score y la curva ROC.

Finalmente, el modelo entrenado se publica en Hugging Face y se le infiere el conjunto de oraciones de Test para que realice la clasificación. (ver bert_classification.ipynb).

Few-Shot

El aprendizaje de pocos ejemplos, también conocido como few-shot learning, es un tipo de aprendizaje automático en el que un modelo es entrenado para generalizar datos a partir de una pequeña cantidad de ejemplos. Es decir, en lugar de entrenarse con miles o millones de ejemplos, un modelo de few-shot learning se entrena con solo unos pocos ejemplos por clase [6].

Para la resolución del problema con esta técnica se utiliza el Framework SetFit proporcionado por Intel Labs, UKP Lab y Hugging Face. SetFit es un marco eficiente para el ajuste fino de Sentence Transformers, que permite alcanzar una alta precisión en tareas de clasificación de texto con pocos datos etiquetados.

El proceso a seguir es el mismo para todos los modelos que se emplean. En primer lugar es necesario la importación del conjunto de los datos para entrenar el modelo. Posteriormente se

descarga el seleccionado empleando el framework SetFit. Llegados a este punto es esencial definir los parámetros que debe tener en cuenta el entrenamiento del modelo, para el problema que se nos plantea de entrenamiento con multietiqueta es estrictamente necesario indicar la estrategia a seguir, en este caso one-vs-rest. Esta estrategia consiste en colocar un clasificador por cada clase, que compara la clase con todas las demás. Es la más utilizada para la clasificación multiclase. Además, se puede utilizar para realizar clasificación multietiqueta, como es el caso. De esta manera, se entrena un clasificador binario de forma independiente para cada etiqueta (método de relevancia binaria) [8].

Respecto a la función de pérdida, se aplica Cosine Similarity Loss. Esta función, se encarga de medir la similitud angular entre dos vectores en un espacio vectorial. Es decir, mide la diferencia entre la similitud coseno deseada y la similitud coseno calculada, y se utiliza para guiar el aprendizaje del modelo durante el entrenamiento [9]. Otros parámetros modificables son el número de épocas que se va a ejecutar el modelo, la cuota de entrenamiento, el tamaño del lote y el número de iteraciones que se ejecuta.

En cuanto a los modelos de Sentence Transformers disponibles en Hugging Face, se aplican los siguientes: all-MiniLM-L6-v2, paraphrase-MiniLM-L3-v2 y paraphrase-mpnet-base-v2, pues se encuentran entre los más descargados. Cabe destacar la eficiencia de los modelos MiniLM, que funcionan hasta 5 veces más rápidos, con menor cantidad de hiperparámetros y manteniendo buenos rendimientos (ver all_MiniLM_L6_v2.ipynb, paraphrase_MiniLM_L3_v2.ipynb y paraphrase_mpnet_base_v2.ipynb) [10].

Por último, mencionar que se ha probado la hiperparametrización de los criterios de entrada del SetFitTrainer sin éxito, debido al alto coste computacional que esto supone y las limitaciones de los recursos que se disponen. A pesar de ello, el proceso a seguir sería el siguiente: inicializar el trainer, seleccionar los posibles parámetros estableciendo rangos u opciones para probar su entrenamiento, y llamar a la función de búsqueda de hiper parámetros que ejecuta un número de iteraciones dado con combinaciones de estos valores. Finalmente, se guardan los valores de los parámetros para la configuración optima y se entrena el modelo con los mismos. Además, existe una función para mostrar una comparativa con los parámetros que más condicionan el entrenamiento del modelo según el conjunto de datos disponibles (ver all_MiniLM_L6_v2_hyperparams.ipynb).

Publicación de modelos en Hugging Face

Otra de las posibilidades que ofrece Hugging Face es la publicación de modelos en su repositorio para que el resto de la comunidad pueda hacer uso de ellos. Esto beneficia a los avances en investigación y la accesibilidad de los recursos producidos por el resto de los desarrolladores [11]. Por ello, se han publicado los modelos entrenados con el conjunto de datos sobre SLAs a Hugging Face. (visitar https://huggingface.co/marmolpen3).

Resultados

En base a las métricas evaluadas, los resultados obtenidos al entrenar distintos modelos no logran alcanzar un valor de Accuracy superior al 55%. A pesar de los potentes algoritmos empleados en el proyecto, la falta de datos y las limitaciones hardware han sido un obstáculo para lograr el rendimiento óptimo en el entrenamiento.

La escasez de los datos se debe a la necesidad de información estructurada del dominio del problema. Ante esa situación, la única alternativa ha sido generar un conjunto de datos que permita al menos, indicarle al modelo cómo debería clasificar. En cuanto a las limitaciones computacionales, los dispositivos que se disponen no permiten entrenar haciendo uso de GPU. Por ello, algunas de las ejecuciones se realizan en las instancias de Colab que ofrece Google. Aun así estas presentan insuficiencia debido al tiempo de ejecución y los límites de recursos en cada cuenta.

Durante el proceso de generación de estos datos, se han probado algunos modelos, siendo la diferencia de los resultados obtenidos al inicio y al final bastante considerable (en torno a un 30% más de accuracy con el doble de datos).

Modelos	**Rendimiento (Accuracy)**
all-MiniLM-L6-v2	0.5294
bart-large-mnli	0.2549
bert	0.5294
paraphrase-MiniLM-L3-v2	0.5490
paraphrase-mpnet-base-v2	0.4313

Se puede observar que el mejor rendimiento se obtiene con el modelo paraphrase-MiniLM-L3-v2 que dispone de la siguiente parametrización: la función de perdida es CosineSimilarityLoss, el tamaño de lote es 12, el número de épocas son 3, las iteraciones 50 y por último la cuota de aprendizaje es de 2e-5.

Conclusiones

El presente trabajo desarrollado en el ámbito del procesamiento de lenguaje natural ha permitido profundizar en una rama muy interesante y que se encuentra en constante evolución. El aprendizaje sobre la biblioteca de Hugging Face presenta la posibilidad de seguir mejorando y aplicando diferentes modelos con mejores ajustes.

A pesar de haber evaluado diferentes alternativas de entrenamiento y modelos, los resultados obtenidos no cumplen con las expectativas deseadas. Es posible que esto sea debido al escaso volumen de los datos disponibles o a la complejidad del problema. Sin embargo, estos resultados no limitan la capacidad de continuar investigando y aplicando nuevas soluciones para mejorar la eficiencia de las predicciones.

El objetivo a futuro es tanto mejorar el conjunto de los datos ampliando su volumen y compartiéndolo con la comunidad para obtener retroalimentación de otros investigadores o evaluación por parte de expertos en el campo que se está tratando, como indagar en los hiperparámetros, la arquitectura y el funcionamiento de los modelos empleados.

Bibliografía

[1] Technical-Committee: “Information technology-cloud computing-service level agreement (SLA) framework-Part 1: Overview and concepts”. Tech. Rep. 19086-1:2016(E), ISO/IEC, Geneva, Switzerland (2016)

[2] Technical-Committee: “CLOUD MODEL SERVICE LEVEL AGREEMENT”. The Common State IT Infrastructure Program("WIIP"). European Commission. 2019

[3] Hugging Face. The hugging face course, 2022. URL https://huggingface.co/course, 2022. [En línea; accedido 03-02-2023]

[4] Hugging Face. Bart model. URL https://huggingface.co/docs/transformers/model_doc/bart. [En línea; accedido 05-02-2023]

[5] Hugging Face. Bert model. URL https://huggingface.co/docs/transformers/model_doc/bert. [En línea; accedido 05-02-2023]

[6] Cloudera, Inc. Few-Shot Text Classification, 2022. URL https://few-shot-text-classification.fastforwardlabs.com/. [En línea; accedido 05-02-2023]

[7] Lewis Tunstall, Nils Reimers, Unso Eun Seo Jo, Luke Bates, Daniel Korat, Moshe Wasserblat, and Oren Pereg. Efficient few-shot learning without prompts, 2022. URL https://arxiv.org/abs/2209.11055.

[8] Scikit-learn developers. Sklearn Multiclass OneVsRestClassifier. 2023. URL https://scikit-learn.org/stable/modules/generated/sklearn.multiclass.OneVsRestClassifier.html#sklearn.multiclass.OneVsRestClassifier [En línea; accedido 05-02-2023]

[9] Google Developers. Keras Loss Cosine Similarity. 2022. URL https://www.tensorflow.org/api_docs/python/tf/keras/losses/CosineSimilarity [En línea; accedido 05-02-2023]

[10] Nils Reimers. Pretrained Models. 2022. URL https://www.sbert.net/docs/pretrained_models.html [En línea; accedido 06-02-2023]

[11] Hugging Face Developers. Sharing pretrained models. 2022 URL https://huggingface.co/course/chapter4/3?fw=pt [En línea; accedido 05-02-2023]