Text Mining and Sentiment Analysis: CTR Classification

Prof. Alfio Ferrara
Arvin Jafari Moghadam Fard

Abstract

This project focuses on using Natural Language Inference (NLI) to help understand clinical trial reports (CTRs), especially for breast cancer. With the explosion of new medical studies, doctors struggle to keep up with the latest information for personalized patient care. Our goal is to create a system that can read and interpret CTRs to determine if a given statement is supported or contradicted by the report. We use data annotated by medical experts to train our system and also explore ways to extract supporting facts to justify its conclusions. This approach aims to make it easier for clinicians to access and use medical evidence effectively.

Introduction

Goals of the Project

Natural Language Inference (NLI) helps computers understand if one statement logically follows from another. For example, if the statement is "He returned to the clinic three weeks later and was prescribed antibiotics," an NLI system should understand that "The patient has an infection" is a logical conclusion.

Recently, the number of Clinical Trial Reports (CTRs) has increased dramatically, making it hard for doctors to keep up with new studies and provide personalized care. NLI can help by quickly interpreting and retrieving important medical information from these reports.

In this project, we focus on using NLI to understand clinical trial data, specifically for breast cancer. We have a dataset of CTRs with statements and labels provided by medical experts. Our goal is to determine if the statements are supported or contradicted by the reports. Additionally, we want the NLI system to find supporting facts from the reports to justify its conclusions.

Dataset

We use a dataset from the Cancer Research UK Manchester Institute and the Digital Experimental Cancer Medicine Team. Each Clinical Trial Report (CTR) has four sections:

1. **Eligibility Criteria**: Conditions for patients to join the trial.
2. **Intervention**: Details about treatments, such as type, dosage, frequency, and duration.
3. **Results**: Information about the trial's participants, outcomes, and results.
4. **Adverse Events**: Symptoms and signs observed during the trial.

Each CTR may have 1-2 patient groups, known as cohorts or arms, which might receive different treatments or have different characteristics.

You can find the dataset and starting kit here.

Baseline

We can use TF-IDF as a good starting point for our task. TF-IDF, which stands for Term Frequency-Inverse Document Frequency, is a way to measure how important a word is in a document compared to a collection of documents. It's widely used in text mining and information retrieval to find the significance of words in large text datasets.

TF-IDF Formula

$$\text{TF-IDF}(t, d, D) = \text{TF}(t, d) \times \text{IDF}(t, D)$$

Here's what the terms mean:

- **TF (Term Frequency)**: This measures how often a term t appears in a document d. It's a way to see how common a word is in that specific document.
- **IDF (Inverse Document Frequency)**: This measures how unique or rare a term t is across all documents in the entire document collection D.

By multiplying TF and IDF, the TF-IDF score tells us how important a word is in a document, considering both its frequency in that document and its rarity across all documents.

Evaluating Text Classification Results Using Performance Metrics

Introduction

In this documentation, we discuss the evaluation of text classification results using key performance metrics, including F1 score, precision, recall, and confusion matrix. These metrics provide insights into the effectiveness and accuracy of classification models, aiding in the assessment of their performance.

F1 Score

The F1 score is a metric that combines precision and recall into a single value, providing a balanced measure of a model's accuracy.

Formula

$$F1 = \frac{2 \times \text{precision} \times \text{recall}}{\text{precision} + \text{recall}}$$

Precision

Precision measures the proportion of correctly predicted positive cases out of all cases predicted as positive by the model.

Formula

$$\text{Precision} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Positives}}$$

Recall

Recall, also known as sensitivity or true positive rate, measures the proportion of correctly predicted positive cases out of all actual positive cases.

Formula

$$\text{Recall} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Negatives}}$$

Confusion Matrix

A confusion matrix is a table that summarizes the performance of a classification model by displaying the counts of true positive, true negative, false positive, and false negative predictions.

Predicted Negative	Predicted Positive
Actual Negative	True Negative (TN)
False Positive (FP)	
Actual Positive	False Negative (FN)
True Positive (TP)	

By utilizing performance metrics such as F1 score, precision, recall, and confusion matrix, we can effectively evaluate the results of text classification models. These metrics provide valuable insights into the model's performance, aiding in the identification of strengths and areas for improvement. Proper evaluation is essential for ensuring the reliability and effectiveness of classification systems in real-world applications.

results on baseline: F1:0.599222 precision$_score$: 0.490446$recall_score$: 0.770000$Aswecannoticetheperforma$
$IDFalogirthm.$

Methodology

Utilizing BERT Embeddings for Text Classification

Introduction

In this documentation, we explore the process of leveraging BERT (Bidirectional Encoder Representations from Transformers) embeddings for text classification tasks. The objective is to utilize the powerful contextual embeddings provided by BERT to enhance the performance of text classification models.

Stage 1: BERT Embeddings Extraction

Overview

BERT, a transformer-based architecture, is utilized to generate embeddings for textual inputs. The pre-trained BERT model is employed to encode input text sequences into fixed-dimensional embeddings.

Implementation

- **BERT Model Selection**: Choose a pre-trained BERT model suitable for the task (e.g., BERT-base, BERT-large).
- **Tokenization**: Convert input text into tokenized format compatible with BERT tokenizer.
- **Embeddings Extraction**: Utilize the BERT model to obtain contextual embeddings for each tokenized input sequence.

Stage 2: Cosine Similarity Analysis

Overview

Evaluate the effectiveness of BERT embeddings using cosine similarity. Cosine similarity measures the similarity between two vectors by calculating the cosine of the angle between them.

Implementation

- **Similarity Calculation**: Compute cosine similarity between pairs of BERT embeddings.
- **Thresholding**: Set a threshold value to classify similarity scores into relevant categories (e.g., similar, dissimilar).
- **Evaluation Metrics**: Measure the performance of cosine similarity analysis using relevant evaluation metrics (e.g., accuracy, precision, recall).

Stage 3: Neural Network Model Training

Overview

Develop a neural network model for text classification using BERT embeddings as input features. Train the model on labeled data to learn the relationship between input embeddings and corresponding class labels.

Implementation

- **Model Architecture**: Design a neural network architecture suitable for the classification task (e.g., CNN, LSTM, Transformer).
- **Data Preprocessing**: Prepare the labeled dataset by encoding class labels and organizing input embeddings.
- **Model Training**: Train the neural network model using the prepared dataset and appropriate training techniques (e.g., batch training, optimization algorithms).
- **Evaluation**: Assess the performance of the trained model using standard evaluation metrics (e.g., accuracy, F1-score) on a separate validation or test dataset.

Addressing Input Length Limitations in BERT-based Text Classification

Introduction

This documentation tackles input length limitations when using BERT models for text classification. We explore methods to overcome this constraint, focusing on text and input chunking techniques to process longer inputs while maintaining classification accuracy.

Problem Statement

BERT's input sequence length limitations pose challenges, particularly with longer textual inputs, impacting the effectiveness of BERT embeddings for comprehensive text understanding in classification tasks.

Proposed Solution

To mitigate input length restrictions, we employ text and input chunking methods. This involves breaking down longer inputs into manageable segments that fit BERT's input length, ensuring all relevant information is effectively processed.

Implementation

Text and Input Chunking

- Use text and input chunking to divide longer inputs into manageable segments.
- Segment text based on predefined chunk sizes to fit BERT's input constraints.

Aggregation Strategy

- Aggregate embedded chunks using a simple averaging approach.
- Compute average embedding vectors for each chunk to represent the entire input sequence.

Benefits

- Enables processing of longer inputs exceeding BERT's input limitations.
- Preserves contextual and semantic meaning by capturing all relevant segments.
- Facilitates BERT embeddings' use in text classification with extensive inputs.

By adopting text and input chunking, we address BERT's input length limitations effectively, ensuring accurate processing of longer inputs for real-world classification tasks.

Results with using bert encoding of input texts: F1:0.479592 precision_score : $0.489583 recall_score : 0.470000$

Dataset Structure Documentation

Introduction

This documentation outlines the structure of the dataset used for training and evaluating text classification models. The dataset comprises three columns, each containing specific information essential for classification tasks.

Dataset Structure

Due to using Siamese Neural Networks, I have structured the dataset in a pairwise input format:

- **Primary_Statement Embeddings**: This column contains pairs of embedded vectors representing the primary sections of statement pairs. Each row represents a pair of embedded vectors corresponding to the primary sections of statements.

- **Secondary_Statement Embeddings (Optional)**: If applicable, this column contains pairs of embedded vectors representing the secondary sections of statement pairs. Similar to the primary section, each row represents a pair of embedded vectors for the secondary sections.
- **Labels**: This column contains the ground truth labels for each statement pair, indicating the correct classification or category assignment. Each label corresponds to a statement pair and is used for training and evaluating classification models.

Siamese Networks: Overview and Application in Text Classification

Introduction

Siamese networks are a class of neural networks designed to compare and analyze similarity between pairs of inputs. These networks learn to generate embeddings for input data in such a way that similar inputs are mapped closer together in the embedding space. In this documentation, we provide a brief explanation of Siamese networks and discuss their application in text classification tasks.

Siamese Networks

Siamese networks consist of twin subnetworks sharing the same architecture and parameters. Each processes a pair of inputs to generate embeddings, which are compared to measure similarity or dissimilarity.

Application in Text Classification

Siamese networks excel in tasks like similarity detection, paraphrase identification, and authorship attribution by encoding semantic information into embeddings. Trained on pairs with labels (similar or dissimilar), they distinguish text pairs and predict their similarity during inference.

Advantages

- Learn complex relationships between inputs, ideal for tasks with subtle semantic differences.
- Effective with limited labeled data by leveraging inherent input similarities.
- Flexible architecture adaptable to various input types and tasks.

Siamese networks provide a powerful framework for text classification, generating meaningful embeddings to discern between similar and dissimilar inputs.

Loss Function

The contrastive loss minimizes distances between embeddings of similar pairs and maximizes those of dissimilar pairs:

- Compute Euclidean distance between embeddings for each pair.
- Minimize distance for positive pairs (same label), maximizing for negative pairs (different labels).
- Aggregate batch losses to derive the contrastive loss.

Results:

- Best Threshold: 0.938
- Best F1 Score: 0.667
- Best Accuracy: 0.510

Discussion and Future Work

In evaluating our Siamese network for text classification, several factors affect our results. A significant challenge is the task's complexity combined with a small dataset. Effective model training for nuanced distinctions in text classification requires diverse data, and our dataset's limited size may lead to suboptimal performance.

The effectiveness of a Siamese network depends heavily on data quality and quantity. In scenarios with limited data, strategies like data augmentation or transfer learning could enhance results.

Handling datasets with secondary statements is crucial. Integrating these statements into the model architecture or processing them separately could significantly improve classification performance.

Future work includes exploring methods like embedding labels with input data for added context and accuracy. Advanced techniques for integrating entire text samples or improving contextual understanding within Siamese networks hold promise for substantial performance gains.

In conclusion, while initial results may not meet expectations, they identify areas for innovation. Addressing data limitations, optimizing model architectures, and exploring new approaches aim to enhance Siamese networks' performance in complex text classification tasks.