Technical Documentation
Extractive Question Answering System Based on BERT

Contents

1 Introduction

This project aims to build an extractive Question Answering (QA) system that can answer questions by extracting the most relevant fragment from a given text. The system uses modern Natural Language Processing technologies based on Transformer models, especially BERT.

The input of the system is:

- A text passage (context)
- A question related to the text

The output is:

- The exact answer span extracted from the text

2 What We Want to Do

The goal of the project is to create a system that:

- Understands a question and a related text
- Finds the most relevant part of the text
- Returns the answer as a text fragment

The system follows a standard NLP pipeline:

1. Data loading
2. Preprocessing
3. Model training
4. Evaluation
5. Inference

3 Starting Point

The project started from existing and well-known resources:

- The BERT architecture
- Hugging Face tutorials and examples
- Pre-trained BERT models
- The SQuAD dataset for Question Answering

These resources provided a solid base for building an extractive QA system.

4 Hugging Face Transformers

The Hugging Face Transformers library provides:

- Standardized implementations for Transformer models (BERT, RoBERTa, DistilBERT, GPT, etc.)
- Optimized tokenizers
- Classes for training and evaluation
- Ready-to-use pipelines for inference

Advantages:

- Allows fast loading of pre-trained models
- Simplifies the fine-tuning process
- Offers a large ecosystem of models and datasets

In the project, Hugging Face is used for:

- Loading the BERT model
- Tokenizing text
- Training and evaluation
- Running inference through the QA pipeline

5 spaCy

spaCy is a natural language processing library focused on real-world applications.

Main features:

- Tokenization
- Sentence segmentation
- Lemmatization
- Syntactic parsing
- Named Entity Recognition (NER)

Characteristics:

- Very fast and optimized

- Provides pre-trained models for multiple languages
- Easy to integrate into applications

In the project, spaCy is used for:

- Cleaning and normalizing text
- Removing invalid characters
- Light processing without altering text positions (important for extractive QA)

6 NLTK (Natural Language Toolkit)

NLTK is one of the oldest and most well-known NLP libraries in Python.

Features:

- Tokenization
- Sentence segmentation
- Stemming and lemmatization
- Stopwords
- Linguistic resources

Advantages:

- Very good for educational purposes
- Offers many classical NLP algorithms

In the project, NLTK is used for:

- Segmenting context into sentences
- Analysis and debugging
- Possible splitting of long texts into chunks

7 scikit-learn

scikit-learn is a classical machine learning library for Python.

It provides:

- Classification, regression, and clustering algorithms
- Evaluation tools
- Data preprocessing utilities

In the project, scikit-learn is used optionally for:

- Computing additional metrics
- Statistical analysis of results
- Performance reports

8 Hugging Face Datasets

Hugging Face Datasets is a library for efficient handling of large datasets.

Features:

- Fast loading of standard datasets (SQuAD, GLUE, etc.)
- Support for map, filter, shuffle operations
- Integration with Transformers

In the project it is used for:

- Loading the SQuAD dataset
- Preprocessing and tokenizing data
- Splitting into training and validation sets

9 Hugging Face Evaluate

Evaluate is a library for computing standard metrics.

Features:

- Implementations for NLP metrics
- Support for SQuAD, BLEU, ROUGE, etc.

In the project it is used for:

- Computing Exact Match and F1 metrics for QA

10 Reused Functionalities

Several components were reused from existing sources:

- Pre-trained BERT models from Hugging Face
- Tokenizers provided by Hugging Face
- Training utilities such as the Trainer class
- The SQuAD dataset format
- Standard QA evaluation metrics (Exact Match, F1)

These elements allowed faster development and reliable performance.

11 Installation

To run the project, the following steps are required:

1. Install Python (version 3.9 or higher recommended)
2. Install required libraries:

```
pip install transformers datasets evaluate spacy nltk scikit-learn torch
python -m spacy download en_core_web_sm
```

12 How to Run

The project supports two main modes: training and inference.

12.1 Training and Evaluation

To train the model on the SQuAD dataset:

```
python main.py train
```

This will:

- Load and preprocess the data
- Train the BERT model
- Evaluate it on validation data
- Save the trained model

12.2 Inference

To use a trained or pre-trained model:

```
python main.py infer --question "Your question" --context "Your text"
```

This will output the extracted answer.

13 Our Added Value

Compared to the original sources and tutorials, this project adds:

- A complete and clean pipeline from data loading to inference
- Integration of spaCy and NLTK for text preprocessing
- A user-friendly command-line interface
- Clear separation between training and inference modes
- Structured code for educational and practical use

14 Project Structure

The logical structure of the project is:

- Data Loading
- Preprocessing (spaCy, NLTK, cleaning)
- Tokenization
- Model Training (BERT)
- Evaluation (Exact Match, F1)
- Inference

All components are integrated in a single main script for simplicity.

15 Example Run

Example:

Context: The book was written by Alice in 2019.

Question: Who wrote the book?

Output:

```
Answer: Alice
Score: 0.98
```

16 Results

After training on the SQuAD dataset, the model typically achieves:

- Exact Match: approximately 70–80%
- F1 Score: approximately 80–90%

These results show that the system performs well for extractive Question Answering tasks.

17 Conclusion

This project demonstrates how modern NLP technologies can be combined to build a functional extractive Question Answering system. By starting from strong existing resources and adding a structured pipeline, preprocessing steps, and user-friendly execution, the project provides both good performance and clear usability.