Tuijin Jishu/Journal of Propulsion Technology
ISSN: 1001-4055
Vol. 46 No. 3 (2025)

Clinical Bert fine tuning for question answering in ICD coding

Nazeer Albokai

Syrian virtual university, Syria

Dr.Bassel Alkhatib

Damascus university, Syria

Dr.Khaled Omar

Damascus university, Syria

Abstract:

In this paper we will propose a method for clinical Bert model fine tuning for questions answering task which utilized in the international classification of disease (ICD) coding process, the fine-tuning process enhance the accuracy of any model since the models are trained on generic data for generic tasks, we used medical clinical Bert model for fine-tuning it on customized medical dataset, the preparation of the training data set is done in many steps to be converted into question/answering suitable format, that the training data set converted to have positive answers and negative answers mapped to the ICD codes, the original data set was contains records of patient claims textual data with the mapped ICD codes, it was converted in a suitable format for questions answering based on the ICD catalog, the data conversion done by utilizing biomedical-ner-all model as medical named entity recognition model to detect the medical entities in training textual data, a training parameters was set to the fine tuning process, before model fine-tuning the accuracy in questions/answering in ICD coding was about 40% , after evaluation our fine-tuned model the accuracy reached to 90%.

Keywords*: biomedical-ner-all, Clinical Bert, Models Fine Tuning, international classification of disease (ICD), ICD coding*

1. Introduction

Question answering is a complicated task in natural language processing field because it needs to understand and analyzing the question before returning the write answer, many algorithms have been developed in question/answering problem, in general we can classify question/answering algorithms according to the methodology to the following:

•Keywords based algorithms: in these types of algorithms the write answer is generated by the keyword's existence, that the algorithm firstly takes the question and search for unique keywords which exist in the question and then by the matching between the question keywords and the answers keywords the final answer is generated.

•Rule based algorithms: in these types of algorithms answer generating is based on a pre-define rules which are updated continuously into the system database.

•Machine learning algorithms: in these types of algorithms answer generating is based on a model, this model already is trained on the questions/answers data set which contains the question and the corresponding answer, in the real time the model is used for inference to predicate the write answer.

•Large language model-based algorithms: in these algorithms finding the right answer for a question is done by the capability of large language models in text generation, there are huge number of large language models which can be utilized in these algorithms.

International classification of diseases (ICD)[1], is a standard for diseases coding and proved by world health organization (WHO)[2], ICD has many releases/versions, ICD coding is essential step in health insurance companies to get the fees of the patients claims, ICD coding process done by specialized persons based on guidelines for choosing the most suitable ICD codes according to patient diagnosis (which written by doctors) ICD coding process contains many errors because of the following [3]:

•maybe doctors use abbreviations in writing diagnosis this in turn leads to incorrect matching in ICD codes.

•in many cases the one diagnosis may relate to more than ICD code.

•in many cases two or three diagnosis may related to the same ICD code.

ICD coding is a hard work and needs a lot of expert manpower resources, so researcher efforts trend to develop a medical models/algorithm that help in make this task automatic, easier and quicker, many medical models have been developed and trained on a medical dataset to help in many tasks in the medical and health sector, we will mention the most popular models in the following:

•Clinical BERT [4]: which was trained on a large multicenter dataset with a large corpus of 1.2B words of diverse diseases, this model can be used for medical text embedding tasks.

•Bio-Medical-MultiModal-Llama-3-8B-V1[5]: is a specialized large language model designed for biomedical applications.

•Mistral-7B Model [6]: This model known for its strong reasoning capabilities and deep language understanding

•Bio-Medical-Llama-3-8B [7]: is a specialized large language model designed for biomedical applications.

•Medical-NER [8]: is finetuned on BERT to recognize 41 Medical entities

Fine tuning [9]is the task of re-training a model on a custom data set to perform a specific task, and it used popular in large language models and small language models.

Fine-tuning is another approach that prioritizes efficiency, and offers several advantages over few-shot learning:

•Quality results: Fine-tuned models generate higher accuracy results.

•Token efficiency: fine-tuned models decrease consumed requests tokens.

The structure of this paper will be, related work, proposed method, evaluation, discussion, conclusion and future work.

2. Related work

Large language models (LLMs)[10] represent one of the most exciting advancements in artificial intelligence in recent years. Built on deep learning techniques—particularly neural networks using the Transformer architecture—these models are trained on vast amounts of data, giving them an impressive ability to understand and generate human-like text. Their development has been a game-changer for natural language processing (NLP), not only enhancing traditional tasks like text generation, translation, and summarization, but also opening doors to new possibilities in areas such as conversational AI, question-answering systems, and sentiment analysis.

The impact of LLMs extends far beyond general applications, revolutionizing specialized fields by boosting efficiency and innovation. In healthcare, for example, these models are proving invaluable in supporting clinical decisions. By processing medical literature, electronic health records (EHRs), and cutting-edge research, LLMs can rapidly suggest diagnoses, treatment options, and medication recommendations—helping doctors make better-informed choices. Researchers have developed several specialized models to refine their medical expertise. Researcher in [11] proposed enhancing existing LLMs by incorporating structured medical knowledge, ensuring responses are grounded in authoritative sources. IvyGPT [12] takes a different approach, blending real-world medical Q&A data with AI-generated content to expand training material. Meanwhile, HuatuoGPT [13] uses

reinforcement learning from AI feedback (RLAIF) to balance insights from both ChatGPT-extracted data and real medical records. Its successor, LLMs also improve patient interactions—powering smart chatbots that answer health questions, schedule appointments, and assist with personal medical records.

In general, the LLM fine-tuning for specialized use cases is shown in the following figure:

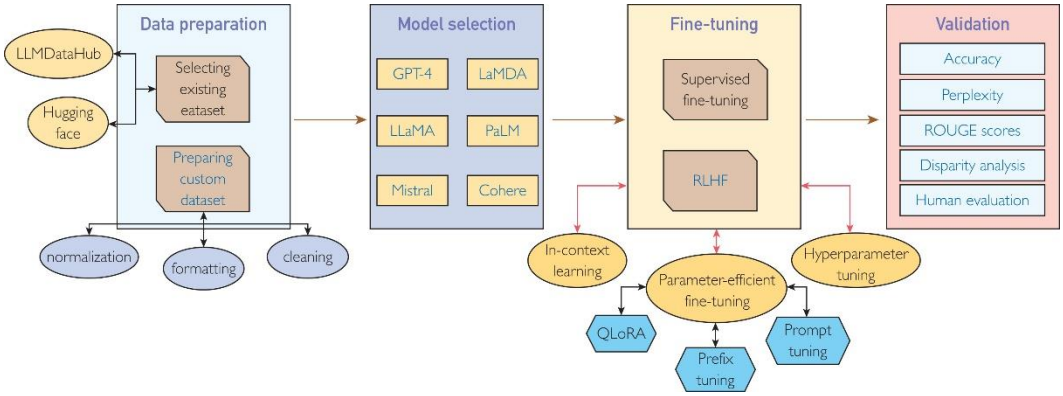

Figure (1) fine tuning workflow [14]

Which contains the following main steps:

•Data preparation: in this step the data is prepared according to the use case that the model will be utilized in, the preparing of data includes a huge step and using of all these step or part of them is depending on the use case, in the following we will mention the most popular techniques in data preparation:

oStop words removing: in this step the useless is removed from the textual data, that each language has a list of common useless words.

oStemming: in this step the words in the textual data are returned to their roots, and the Plural words turned into singular words, the most popular stemmer for English language Stanford stemmer [15].

oSentences detection: in this step the boundaries of the sentences are detected, mainly the sentences detected according to the punctuation marks.

oKeywords extraction: in this step the unique keywords are detected, the keywords detection based on statistical features and on words meaning features.

oTerm Frequency - Inverse Document Frequency (TF/IDF) evaluates the importance of the word in a document or corpus [16], that the TF is measure the important of the term in a one document and IDF is a measure of the importance of the term in all corpus documents.

oN-gram is the contiguous sequence of words, where N may be ‘1’, ‘2’, ’3’, and so on. One word sequence is called 1-gram (or unigram), sequence of two words is called 2-gram (or bi-gram), sequence of three words is called 3-gram (Tri-gram), and so on [17].

•Model selection: in this step the model is selected, that there are a huge number of large language models (generic models, and domain-oriented models), also there are free large language models like Ollama [18] , and paid large language models like ChatGPt.

•Fine-tuning: in this step the hyperparameters and the parameter efficient fine tuning are set to the fine-tuning task, which contains the number of ecphocs,

•Validation: in this step the fine-tuned model is validates on the validation data set to calculates the accuracy of it according to the use case, usually the original data set is divided to training, testing, evaluation data sets.

3. Propsed Method

In this section we will describe in details the workflow of the proposed methodology, the following figure contains the steps of algorithm workflow:

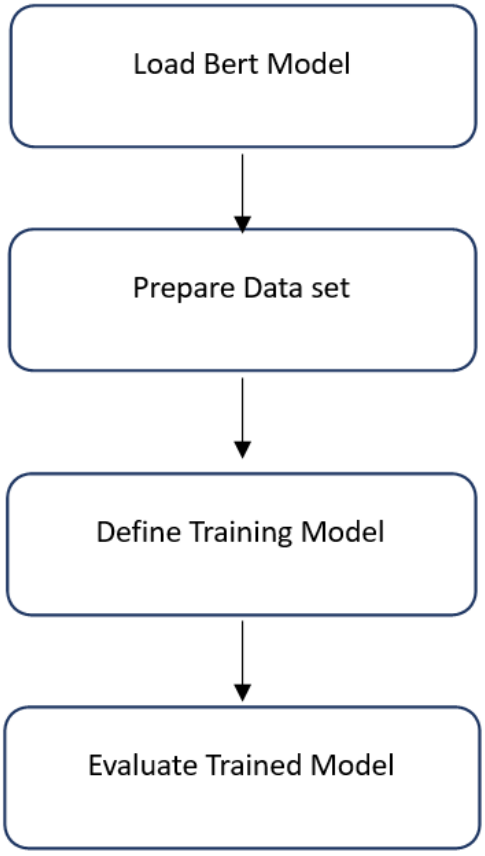

Figure (2): algorithm workflow

3.1. Load Bert Model:

In this step the source model is loaded, we have chosen a Clinical Bert model for fine tuning, the following python code describes the load implementation:

```
model_name = "medicalai/ClinicalBERT"
model = SentenceTransformer(model_name)
```

the model is downloaded from hugging face, so based on the services on hugging face all we need to handle any model just we need to write its name on hugging face cloud.

3.2. Prepare Data Set:

In this step the data set for model fine tuning is prepared, the raw data was as the following structure, it was a records of claim textual data, which contains the claim text and the mapped ICD code with this claim text, the following table shows the raw data structure:

Record Number	Claim textual data	ICD code
1	Text[1]	Code[1]

2	**$Text^2$**	**$Code^2$**
.	**…**	**…**
.	**…**	**…**
N	**$Text^n$**	**$Code^n$**

Table (1): training raw data

After loaded the raw data, we have grouped the rows according to the mapped ICD code, the aim of grouping the data according to unique ICD codes is to remove redundant ICD codes, after this step each ICD code will be mapped to many claims textual data (which have the same ICD code), as the following structure:

Record Number	Claim textual data	ICD code
1	**$Text^1$ $Text^{20}$ ….**	**$Code^1$**
2	**$Text^2$ $Text^{40}$ ….**	**$Code^2$**
.	**…**	**…**
.	**….**	**…**
N	**$Text^{n-1}$ $Text^{30}$….**	**$Code^n$**

Table (2): grouped training data set

Then to remove useless textual information, we will apply medical named entity extraction (NER) on the " Claim textual data" column, to implement this step we have used biomedical-ner-all model [19], this model can extract 107 medical entities and it has been trained onto Maccrobat data set [20], by the end of this step the data will be in the following structure:

Record Number	Claim textual entities	ICD code
1	**$Entity^1$ $Entity^2$ …**	**$Code^1$**
2	**$Entity^{10}$ $Entity^{11}$ …**	**$Code^2$**
.	**…**	**…**
.	**…**	**…**
N	**$Entity^{n-2}$ $Entity^{n-1}$ …**	**$Code^n$**

Table (3): training data set with extracted entities

The following python codes shows how we used biomedical-ner-all model :

```
from transformers import pipeline

from transformers import AutoTokenizer, AutoModelForTokenClassification

tokenizer = AutoTokenizer.from_pretrained("d4data/biomedical-ner-all")

model = AutoModelForTokenClassification.from_pretrained("d4data/biomedical-ner-all")

pipe = pipeline("ner", model=model, tokenizer=tokenizer, aggregation_strategy="simple") # pass device=0 if using gpu

pipe("""The patient reported no recurrence of palpitations at follow-up 6 months after the ablation.""")
```

After this step we will prepare the data set to be in a questions/answers data set, so first we will load the international classification of diseases catalog (ICD10) , which is accredited from world of health organization (WHO), it contains the following columns:

•ICD code: disease code

•Chapter: disease chapter

•Short description: disease short description

•Long description: disease long description

We will choose only two columns (ICD code, long description), then we will merge the selected columns with the data set sheet which shown in table (3), the merging process will be on the ICD code column, so the generate data set from merge shown in the following table:

Record Number	Claim textual entities	ICD code	ICD code long description
1	$Entity^1$ $Entity^2$ …	$Code^1$	$Description^1$
2	$Entity^{10}$ $Entity^{11}$ …	$Code^2$	$Description^2$
.	…	…	…
.	…	…	…
N	$Entity^{n-2}$ $Entity^{n-1}$ …	$Code^n$	$Description^n$

Table (4): entities training data set with code, description

So based on the table above, now each ICD code mapped to n medical entity and long description.

After this step two column will be selected only (Claim textual entities, ICD code long description as positive ICD code description), we exclude ICD code column because our aim to generate a questions/answers data set, so the following table shows the dataset after ICD code exclusion.

Record Number	Claim textual entities	Positive ICD code long description
1	$Entity^1$ $Entity^2$ …	$Positive\ Description^1$
2	$Entity^{10}$ $Entity^{11}$ …	$Positive\ Description^2$
.	…	…
.	…	…
N	$Entity^{n-2}$ $Entity^{n-1}$ …	$Positive\ Description^n$

Table (5): entities training data set with entities, Positive ICD code long description

Record Number	Claim entities	Positive ICD code long description	Negative ICD code long description
1	$Entity^1$ $Entity^2$ …	$Positive\ Description^1$	$Negative\ Description^1$

2	**Entity[10] Entity[11] …**	**Positive Description[2]**	**Negative Description[2]**
.	**…**	**…**	**…**
.	**…**	**…**	**…**
N	**Entity[n-2] Entity[n-1] …**	**Positive Description[n]**	**Negative Description[n]**

Table (6): entities training data set with entities, Positive ICD code long description, Negative ICD code long description

The task of generate negative ICD code description is done as the following for each record from the data set in table (5), negative descriptions will be generated randomly, that the algorithm will assign a random ICD code description for each record, by the end of this step we will have the final data set for the fine tuning as the following:

Now the data set is ready for the model fine tuning, which contains the following columns:

•Record Number: this is the record number

•Claim entities: this column contains the medical entities which extracted from the medical claim textual input, the entities have been extracted by using medical named entities model which is

•Positive ICD code long description: this column contains the right international classification of diseased description

•Negative ICD code long description: this column contains the false international classification of diseased description

Now the data set is ready for question/answer task, this data set is divided as the following:

•80 % as training data set

•10% as validation data set

•10% as test data set

3.3. Define Training Model:

In this step the training parameters are set to the model as the following:

```
train_args = SentenceTransformerTrainingArguments(
    output_dir=f"models/{finetuned_model_name}",
    num_train_epochs=num_epochs,
    per_device_train_batch_size=batch_size,
    per_device_eval_batch_size=batch_size,
    learning_rate=lr,
    warmup_ratio=0.1,
    batch_sampler=BatchSamplers.NO_DUPLICATES,
    eval_strategy="steps",
    eval_steps=100,
```

logging_steps=100)

where:

•num_epochs=60

•batch_size=16

Also, before begin the training phase, the lose function is defined as the following:

loss = MultipleNegativesRankingLoss(model)

and also, the evaluation function is defined as the following:

```
evaluator_valid = TripletEvaluator(
    anchors=dataset["validation"]["Entities"],
    positives=dataset["validation"]["PosLongDesc"],
    negatives=dataset["validation"]["NegLongDesc"],
    name="ai-job-validation",
)
```

then the training phase is called by the following python code:

```
trainer = SentenceTransformerTrainer(
    model=model,
    args=train_args,
    train_dataset=dataset["train"],
    eval_dataset=dataset["validation"],
    loss=loss,
    evaluator=evaluator_valid,)
trainer.train()
```

3.4. Evaluate Trained Model:

In this step the trained model is evaluated, the evaluation done using the following python code:

```
evaluator_test = TripletEvaluator(
    anchors=dataset["test"]["Entities"],
    positives=dataset["test"]["PosLongDesc"],
    negatives=dataset["test"]["NegLongDesc"],
    name="ai-job-test2",)
```

4. Results

4.1. Data set

We have tested our fine-tuned model on dataset which contains 500 records, each record of this data set contains two columns, the first one is the claim textual data, and the second one is the correct ICD code which mapped by clinical experts.

4.2. Evaluation metrics

We have used the following formula to evaluate our fine-tuned model [15]:

$$Precision = \frac{(ExtractedICDcodes \bigcap ICDcodes)}{ExtractedICDcodes}$$

ICD coding Precision formula [21]

Where:

- Extracted ICD codes: are the codes which are generated by using our fine-tuned model.
- ICD codes: are the correct ICD codes which already mapped to the claims textual data.

The evaluation of our fine-tuned model leads to 90% of accuracy in ICD coding.

5. Discussion

In this paper we have enhanced the ICD coding task accuracy , by fine-tuning Clinical BERT model on a medical dataset like as a questions answering dataset that each row in this dataset contains the medical entities, positive description(which is the right ICD code long description), and negative description(which is the wrong ICD code long description), the data set was prepared throw many steps to be converted in the used format, and it was divided to training, validation and test data set, the generated model has been evaluated onto validation and test data set parts, and the results reached to 90% accuracy for the fine-tuned model, and for model inference public usage we have to pass the claim textual input to a medical named entity recognition(NER) model to detect the medical entities into unstructured input text, after that the generated model will return to us the most relevant ICD code long description and from returned text we will get the right ICD code for the input claim, by using this model the ICD coding task will be more easier and more accurate than using any general large language model.

Refrences

[1] International Classification of Diseases (ICD). (n.d.). https://www.who.int/standards/classifications/classification-of-diseases
[2] Home. (2025, June 19). https://www.who.int/
[3] Most common ICD-10 error codes - HIS. (n.d.). Healthcare Information Services. https://healthinfoservice.com/blog/most-common-icd-10-error-codes/
[4] emilyalsentzer/Bio_ClinicalBERT · Hugging Face. (n.d.). https://huggingface.co/emilyalsentzer/Bio_ClinicalBERT
[5] ContactDoctor/Bio-Medical-MultiModal-Llama-3-8B-V1 · Hugging face. (n.d.). https://huggingface.co/ContactDoctor/Bio-Medical-MultiModal-Llama-3-8B-V1
[6] mistralai/Mistral-7B-v0.1 · Hugging Face. (n.d.). https://huggingface.co/mistralai/Mistral-7B-v0.1
[7] ContactDoctor/Bio-Medical-Llama-3-8B · Hugging face. (n.d.). https://huggingface.co/ContactDoctor/Bio-Medical-Llama-3-8B
[8] blaze999/Medical-NER · Hugging Face. (n.d.). https://huggingface.co/blaze999/Medical-NER
[9] Fine-tuning. (n.d.). https://huggingface.co/docs/transformers/en/training
[10] IBM. (2025, May 20). Large language models. LLM. https://www.ibm.com/think/topics/large-language-models
[11] Kai He, Rui Mao, Qika Lin, Yucheng Ruan, Xiang Lan, Mengling Feng, Erik Cambria,A survey of large language models for healthcare: from data, technology, and applications to accountability and ethics,Information Fusion,Volume 118,2025,102963,ISSN 1566-2535,https://doi.org/10.1016/j.inffus.2025.102963.

[12] Duan, Yaofei & Lam, ChanTong & Chen, Jiexi & Xu, Jiangsheng & Chen, Haoming & Liu, Xiaohong & Pang, Patrick Cheong-Iao & Tan, Tao. (2023). IvyGPT: InteractiVe Chinese pathwaY language model in medical domain. 10.48550/arXiv.2307.10512.

[13] Hongbo Zhang, Junying Chen, Feng Jiang, Fei Yu, Zhihong Chen, Guiming Chen, Jianquan Li, Xiangbo Wu, Zhang Zhiyi, Qingying Xiao, Xiang Wan, Benyou Wang, and Haizhou Li. 2023. HuatuoGPT, Towards Taming Language Model to Be a Doctor. In Findings of the Association for Computational Linguistics: EMNLP 2023, pages 10859–10885, Singapore. Association for Computational Linguistics.

[14] D.M. Anisuzzaman, PhD · Jeffrey G. Malins, PhD · Paul A. Friedman, MD · Zachi I. Attia, PhD,Fine-Tuning Large Language Models for Specialized Use Cases,Mayo Foundation for Medical Education and Research., November 28, 2024,DOI: 10.1016/j.mcpdig.2024.11.005

[15] Stemming and lemmatization. (n.d.-b). https://nlp.stanford.edu/IR-book/html/htmledition/stemming-and-lemmatization-1.html

[16] Nkhoma, K. (2024, November 17). TF-IDF (Term Frequency–Inverse Document Frequency) - AIOSEO. All in One SEO - Best WordPress SEO Plugin. https://aioseo.com/seo-glossary/tf-idf-term-frequency-inverse-document-frequency/

[17] Patil, S. (2024, November 22). Understanding N-Grams: The Foundation of Language Modeling in NLP. Medium. https://medium.com/@sonalpatil1810/understanding-n-grams-the-foundation-of-language-modeling-in-nlp-f9f632459519

[18] Ollama. (n.d.). Ollama. https://ollama.com/

[19] d4data/biomedical-ner-all · Hugging Face. (n.d.). https://huggingface.co/d4data/biomedical-ner-all

[20] MACCROBAT. (2023, December 15). Kaggle. https://www.kaggle.com/datasets/okolojeremiah/maccrobat

[21] Omar, Khaled & Kawas, Mohamad & Alkhatib, Bassel. (2024). Multi-Layer Algorithm for ICD prediction. 20. 394-404