SNOMED CT Entity Linking Challenge: MITEL-UNIUD Solution Documentation

1. Team members

Mihai Horia Popescu: PhD Candidate, University of Udine, Italy.
Member of the MITEL lab, **Mihai** is a PhD student at the University of Udine, North-East of Italy. His research interests include Explainable artificial intelligence, Natural Language Processing, Computer Vision, Data mining and analysis, Machine Learning, and Statistical Modeling over Medical Domain. He collaborated with multiple organizations in the domain of mortality and morbidity, where he worked on the development of tools supporting automated coding.

Kevin Roitero: Tenure Track Assistant Professor, University of Udine, Italy.
His research interests include Artificial Intelligence, Crowdsourcing, and Information Retrieval. He visited and collaborated with multiple top universities across the globe as well as with leading industry partners, publishing papers in top ranked and selective conferences and in top-tier journals. As result of his work, he received multiple grants and awards, including the participation in the 7th edition of the prestigious Heidelberg Laureate Forum (top 200 young researchers in mathematics and computer science).

Vincenzo Della Mea: Associate Professor, University of Udine, Italy.
Vincenzo is the head of the MITEL lab and professor of Medical Informatics in both the Computer Science and Medicine degrees. His research is focused on biomedical classifications and terminologies on one side, and image analysis on histopathologic images on the other side. In the last years he explored the use of deep learning techniques in both areas. He is associate editor of Digital Health and of the International Journal of Telemedicine and Applications.

2. Motivation

SNOMED-CT and the International Classification of Diseases (ICD) are pivotal in clinical terminology and classification. Leveraging our experience with ICD automated coding, we aim to focus on SNOMED-CT by engaging in this competition. The provided data will significantly aid our research in automated clinical document coding and annotation, aligning with the interests of the Challenge sponsors and demonstrating the primary application of SNOMED-CT.

3. Summary of our Approach

Our approach to the challenge involves two primary tasks: first, an *entity recognition* task aided by a Large Language Model (LLM), which involves annotating terms within the input text by means of a specific prompt tailored on annotating clinical entities. Secondly, we implement classification in two stages: an initial *document retrieval* task carried out using the vector database FAISS, followed by a *classification* task performed leveraging an LLM, again with a specific prompt.

Entity Recognition

Initially, we split the text of each note into chunks to allow our chosen model, `mistralai/Mistral-7B-Instruct-v0.2`, to extract terms in a narrow and more context-homogeneous text. The model was fine-tuned on part of the training set. We also experimented with longer texts but we did observe a decline in entity recognition effectiveness. Our optimal strategy involves using two models: the former processing chunks with a length of 100 tokens, and the latter processing chunks of 500 tokens. After splitting the text into chunks, we apply these fine-tuned models on each segment to annotate and identify terms (in terms of span of text). Post-annotation, we combine the results of the two models, resolving potential overlaps by retaining the longer annotation from the two models. This dual-model approach yielded better effectiveness compared to a single-model application.

SNOMED-CT Coding

In our classification framework, a pivotal role is played by the integration of `Faiss` for document retrieval, which includes the index for all relevant SNOMED-CT terms and their synonyms as pertinent to the challenge. We augmented the `Faiss` index with pairs of annotations, creating a comprehensive dictionary that tracks term usage and their corresponding codes, acknowledging that a single term may have multiple occurrences and be annotated with different codes based on its context. This enriched vector database has been used to retrieve the top 10 text chunks, which then informed the subsequent classification phase.

For the fine-tuning process, we employed a `Mistral` model that incorporates various contextual elements: the term to be classified itself, its surrounding text used to provide context, the section title, and the top 10 text chunks as retrieved from the Faiss vector database. We found this multifaceted approach to have higher classification effectiveness for the final code attributed to each term.

Moreover, to further enhance the results, our methodology included a "remove list" and an "add list" set of terms that have been used to refine the entity recognition output. The “remove list” set excludes terms that, while suggested by the model, are either irrelevant or not part of the training dataset, according to a dictionary-based analysis of the training corpora. This exclusion list addresses issues like stop-words and other non-annotatable but frequently appearing sets of terms. Conversely, the “add list” includes terms that, though consistently overlooked by the model, should always be annotated due to their significance.

The embedding model employed to build the Faiss vector database is `sentence-transformers/all-MiniLM-L12-v2`, chosen for its effectiveness in generating meaningful vector representations that facilitate accurate document retrieval and subsequent classification.

Open source status

In the development of our solution, in addition to the usual software (Python, etc) we exploited the following external components, which are all open source:

- `Mistral`: Apache 2.0 license

- `Faiss`: MIT license
- `all-MiniLM-L12-v2`: Apache 2.0 license
- `vLLM`: Apache 2.0 license
- `huggingface/peft`: Apache 2.0 license

4. Charts

We don’t have any informative graphs or charts.

5. Impactful parts of code

Some of the most impactful parts of the proposed solution are the prompts used by the models and the Faiss code, both reported in the following.

Prompt template for inference

```
[INST]You are a medical practitioner tasked with classifying a discharge
note for a patient. You have a hospital discharge note and need to
correctly identify and annotate the medical terminologies based on specific
instructions.

Instructions:
- The discharge note has informations about a patient discharge, and
sections can be identified with the followed regex: "^\s*\w+:".
- The terminologies are annotated using tags: <t> to start the annotation
and </t> at the end of the annotation.
- '___' are parts of the note that have been anonymized. Ignore them.
- The terms of interest can be synonyms and definitions used in clinical
documentation.
- Below, you are given the categories of a specific classification which
you need to identify.
    - Medical Conditions and Disorders:
        - Includes general and specific health conditions, such as heart
function, respiratory rate, mental status, kidney function, inflammation,
and various system-specific conditions (e.g., cardiovascular, pulmonary).
    - Diagnosis and Clinical Findings:
        - Encompasses various aspects of patient assessment, like
discomfort, body temperature, mortality, vital signs, and functional
status.
    - Treatment and Management:
        - Covers a range of treatments, including rehabilitation
guidelines, pain management, surgical procedures, fluid and electrolyte
replacement, nutritional regimens, and medication management.
    - Healthcare Services and Processes:
        - Encompasses aspects of healthcare delivery, such as care plans,
medical consultations, healthcare decision-making, post-hospitalization
instructions, and review of treatment progress.
    - Lifestyle and Supportive Care:
        - Focuses on patient lifestyle factors and supportive care,
including diet, hydration, personal hygiene, physical activity, and the
ability to perform daily activities.
    - Surgical and Recovery Aspects:
        - Includes terms related to surgical procedures, wound healing,
recovery units, and post-surgical care.
    - Monitoring and Evaluation:
        - Encompasses terms related to the ongoing monitoring and
evaluation of patient health, such as cardiac output, fluid intake, and
urination.
- Annotate only the terms that can be classified under the list and not the
entire phrase where you identify it.
- Be very precise, every character matters!
- Please provide a brief and direct answer, avoid any extraneous details or
explanations to align with the specific context and keywords!
- Section titles need to be ignored.
- If no suggestions, reply with the input text and no annotations.
- You must always respond with the original note with the terminologies
annotated.
# Hospital discharge note:
{query} [/INST]
```

Prompt template for classification

```
[INST]You are presented with a specific medical term that requires
classification. Below is a list of the top-10 most similar classification
codes, obtained through a FAISS vector search. Each code is provided
alongside a description to assist in your selection process. Examine the
options carefully and select the number (ranging from 0 to 9) that
accurately represents the correct classification code for the term in
question.

Term to Annotate: {term}
Section name: {section}
Context of the term: {context}

Classification Codes:
{faiss}

Instructions:
Review the classification codes and their descriptions carefully.
Determine which number (0-9) best matches the classification code for the
provided term.
Reply -1 if none is correct.
Submit your response by entering the selected number only. Please refrain
from adding any additional text, comments, or explanations.
Respond Below:[/INST]
```

Code assignment

Code assignment using FAISS is carried out using of the following code

```
FAISS_DB = LoadVectorize.load_db(faiss_index, model_path_faiss,
model_path_faiss_cache)
    df_concepts = LoadVectorize.load_dataset_synonyms(terminologies_path)
    df_dict = df_concepts.set_index('concept_name_clean')['concept_id'].to_dict()
nr_threads = os.cpu_count()
```

```
for index_o, row in df_notes.iterrows():
        if multiprocessing:
            futures = concurrent_calls(FAISS_DB, df_dict,
row['result_chunks_inst'])
            for index_i, res_pair in enumerate(futures):
                top_docs = res_pair[1]
                df_notes.at[index_o, 'result_chunks_inst'][index_i] =
list(df_notes.at[index_o, 'result_chunks_inst'][index_i])
                df_notes.at[index_o, 'result_chunks_inst'][index_i][3] =
top_docs[0][0]
                df_notes.at[index_o, 'result_chunks_inst'][index_i][6] =
','.join([str(val) for val, score, doc in top_docs])
                df_notes.at[index_o, 'result_chunks_inst'][index_i][7] =
','.join([str(score) for val, score, doc in top_docs])
                df_notes.at[index_o, 'result_chunks_inst'][index_i][8] =
','.join([doc for val, score, doc in top_docs])
        else:
            for index_i, term in enumerate(row['result_chunks_inst']):
                top_docs = do_search(FAISS_DB, df_dict, term)[1]
                df_notes.at[index_o, 'result_chunks_inst'][index_i] =
list(df_notes.at[index_o, 'result_chunks_inst'][index_i])
                df_notes.at[index_o, 'result_chunks_inst'][index_i][3] =
top_docs[0][0]
                df_notes.at[index_o, 'result_chunks_inst'][index_i][6] =
','.join([str(val) for val, score, doc in top_docs])
                df_notes.at[index_o, 'result_chunks_inst'][index_i][7] =
','.join([str(score) for val, score, doc in top_docs])
                df_notes.at[index_o, 'result_chunks_inst'][index_i][8] =
','.join([doc for val, score, doc in top_docs])
        print("Document nr: " + str(index_o) + " assigned")
```

6. Machine Specs

Our solution was implemented on a server running Ubuntu, equipped with the following hardware specifications:

- CPU: Intel(R) Core(TM) i7-10700 CPU @ 2.90GHz, featuring 8 cores and 16 threads.
- GPU: NVIDIA RTX A6000.
- Memory: 64GB of RAM.
- Operating System: Ubuntu 22.04.

The training duration has been approximately 6 hours and the inference duration has been approximately 40 minutes on the 54 notes of our test set.

7. Special requirements

We enhanced our methodology by integrating vLLM, a library optimized for efficient LLM serving and inference. vLLM boosts performance through advanced features like PagedAttention, continuous request batching, and optimized CUDA kernels. It supports various Hugging Face models, offers high-throughput serving with different decoding algorithms. Its design facilitates fast LLM inference, essential for our project's requirements.

8. Exploratory Data Analysis and Data Preparation

Initially, we extracted spans and corresponding SNOMED-CT codes for analysis. To enhance our dataset, we developed a SNOMED-CT vocabulary, starting from a flattened version of the terminology provided and further enriching it. Our data analysis involved inspecting and refining the training annotation spans to increase their homogeneity to aid the model. Specifically, we eliminated annotations that consisted solely of spaces and non-alphanumeric characters, and we adjusted certain annotation spans to exclude trailing or leading spaces and corrected instances where not all characters of a terminology term were included within the span.

9. Performance Evaluation

We conducted local evaluations using different methodologies to assess our entity recognition and classification components.

For entity recognition, we divided the set of 204 clinical notes into 150 for training and 54 for testing. We evaluated the precision, recall, and F1 measure per note by comparing the extracted terminologies with gold standard annotations, disregarding the positions of the spans. Additionally, we computed precision, recall, and F1 scores by matching the identified spans with gold annotations without considering the term classification, using a unique identifier from the concatenation of "note_id," "start," and "end."

In the classification evaluation, we assessed the FAISS retrieval using the first 150 notes' annotations for training and the remaining 54 for testing. We determined the classification accuracy based on the correctly annotated spans from entity recognition, examining FAISS's retrieval performance at various levels (@1, @5, @10, and @20). For the final classification step, we used an entity recognition model trained on 150 notes and a FAISS index from these notes. The model was then trained on 30 notes and tested on 24, focusing on the classification accuracy @1 post-model classification with the retrieved documents, terms, their sections, and phrase context. We also employed Intersection over Union (IoU) for further evaluation, providing a comprehensive assessment of our entity recognition and classification approach's joint effectiveness.

Finally, we did some qualitative evaluation by looking at predicted codes frequencies vs ground truth frequencies.

10. Quick overview of collateral experiments

While initially setting up our approach and environment, we implemented a dictionary-based solution to establish a baseline, achieving scores of 0.3619 on the public leaderboard and 0.32 on the private one. Subsequently, we transitioned to our main solution, exploring various entity recognition strategies using a LLM.

Our first strategy involved generating all possible terms as output without the model identifying specific spans, mapping these results to their corresponding text chunks. This method provided high recall but suffered from low precision due to the identification of multiple terms in the same chunk, not all of which were relevant according to the gold annotations.

The second strategy focused on generating the input text with annotations inserted around terms, such as <t>term</t>. However, this method did not always accurately reproduce the full input, necessitating the regeneration of poorly generated text chunks. We considered enhancing this strategy with a Named Entity Recognition (NER) task but did not have the opportunity to test this improvement.

Lastly, we experimented with a hybrid approach, generating only the relevant terms and identifying their positions by annotating the preceding word. If a term was part of a larger word, we generated the entire word and used braces to indicate the term, for example, start{term}end.

Through these methods, we refined our entity recognition process, with each strategy offering unique insights and contributing to the evolution of our final solution.

11. Further developments

For future work, it would be beneficial to explore the use of a LLM specifically trained for NER tasks rather than relying on text generation for entity recognition. This shift could enhance the precision and effectiveness of entity identification within texts. Additionally, for the classification task, expanding the dataset to include additional synonyms for SNOMED-CT entities could prove advantageous. Currently, utilizing only the official synonyms from the classification, FAISS @20 achieves an accuracy of 0.65. However, by incorporating training annotations, the accuracy at @20 improved significantly to 0.90 using the 150/54 split. This suggests that enriching the synonym dataset could further refine the classification accuracy, making it a promising direction for enhancing the system's overall performance.
Aside from our possible developments, we think that IoU, as is, is not the best metrics to capture the complexity of SNOMED-CT coding. SNOMED-CT is a large terminology, with no specific rules on the amount of annotations to be carried out in a clinical document. Thus, annotation can be more or less complete, without being for this more or less wrong. IoU tends to penalize a system that annotates more than the human annotators, but the additional annotations are not necessarily wrong - sometimes redundant, sometimes just spot on, even if not manually annotated. However, it is not easy to imagine a different metric for this, it is something that needs some study.

12. Optimizations

In our ongoing efforts to optimize the system, one potential improvement we're considering is experimenting with quantized models, specifically 8-bit and 4-bit versions. These quantized models offer the advantage of reducing memory consumption, particularly in terms of saving vRAM. However, it's important to note that while these models can be more efficient in terms of memory usage, there is a potential trade-off in terms of accuracy. Therefore, while the prospect of quantized models might lead to memory efficiency, it's crucial to thoroughly test these models to assess their impact on accuracy.