Model documentation and write-up

1. Who are you (mini-bio) and what do you do professionally? If you are on a team, please complete this block for each member of the team.

   We are a team of researchers from the KI research institute in Israel, a non profit promoting applied research in computational health. Guy, Yonatan and Chen received their PhD in computer science some 20 years ago, while Irena is catching up to them these days. Before joining the KI research institute Guy, Yonatan and Chen worked in the Haifa IBM Research lab on machine-learning projects; Guy on medical imaging, Yonatan on computational argumentation and Chen on analysis of electronic health records. Irena used to work for Medial Research, a company developing machine-learning solutions for early detection of disease,

2. What motivated you to compete in this challenge?

   We have always been interested in leveraging NLP tools for benefit in the medical domain. This became especially attractive with recent advancements, and in particular the rapid improvements in LLMs. We hoped that by participating in the challenge we will become familiar with the current tools, and be able to evaluate their usefulness.

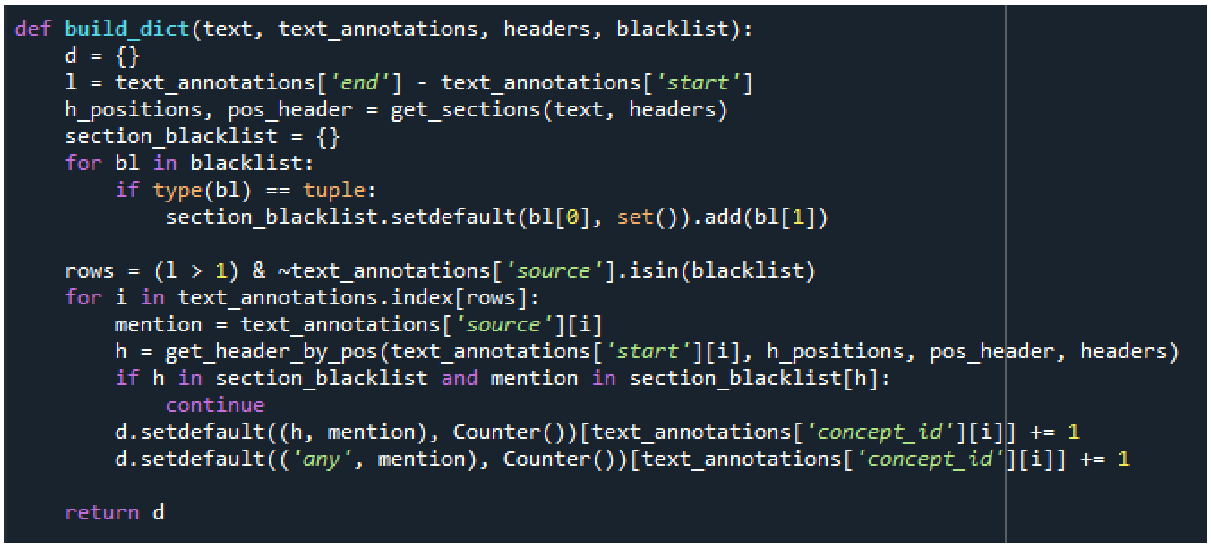
3. High level summary of your approach: what did you do and why?

   Initially we aimed to develop an LLM-based solution, but as a baseline for comparison we also tried a simple dictionary-based approach. We proceeded in parallel, but fairly early on it seemed that in the context of this challenge, with the available annotated data, the dictionary approach was more promising. This shifted our focus to an iterative process of error analysis and improving the dictionary construction algorithm.

4. Do you have any useful charts, graphs, or visualizations from the process?

   From the onset we have built an html-based solution for visualization of annotations and comparing them. Having a custom-made tool allowed us to modify it as we proceeded, according to the arising needs.

5. Copy and paste the 3 most impactful parts of your code and explain what each does and how it helped your model.

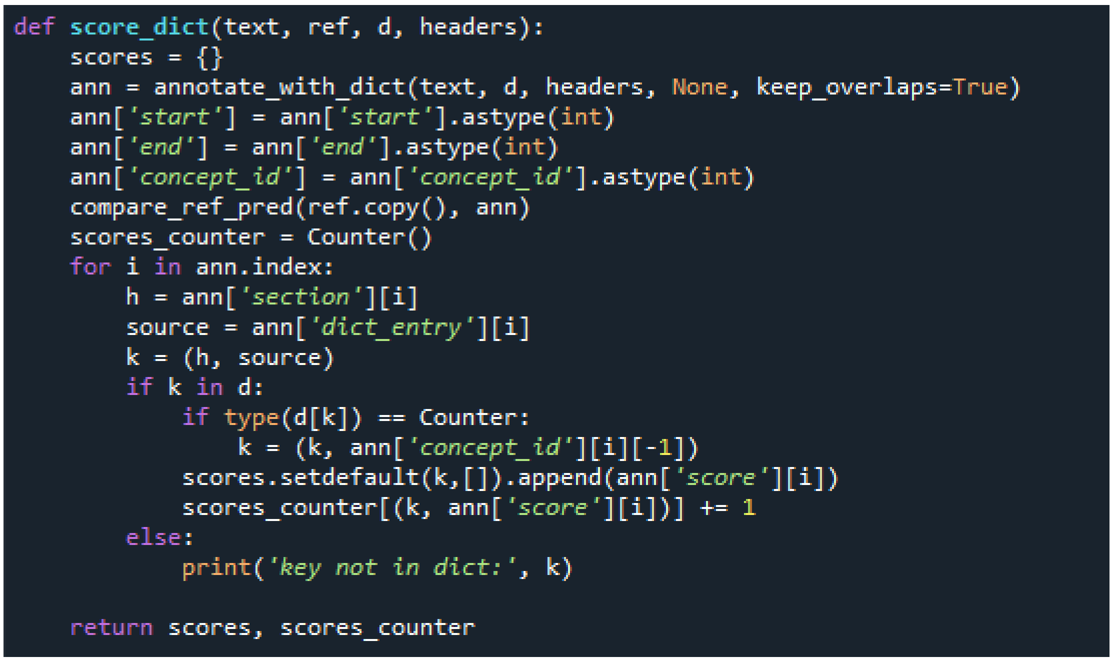


```python
def build_dict(text, text_annotations, headers, blacklist):
    d = {}
    l = text_annotations['end'] - text_annotations['start']
    h_positions, pos_header = get_sections(text, headers)
    section_blacklist = {}
    for bl in blacklist:
        if type(bl) == tuple:
            section_blacklist.setdefault(bl[0], set()).add(bl[1])

    rows = (l > 1) & ~text_annotations['source'].isin(blacklist)
    for i in text_annotations.index[rows]:
        mention = text_annotations['source'][i]
        h = get_header_by_pos(text_annotations['start'][i], h_positions, pos_header, headers)
        if h in section_blacklist and mention in section_blacklist[h]:
            continue
        d.setdefault((h, mention), Counter())[text_annotations['concept_id'][i]] += 1
        d.setdefault(('any', mention), Counter())[text_annotations['concept_id'][i]] += 1

    return d
```

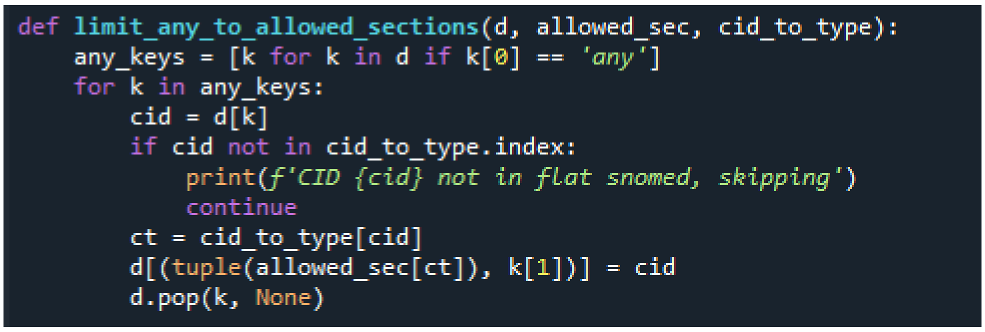
This function builds the initial dictionary. Importantly, it considers the section in which a mention was found and uses it as context for later inference. Also, any mention can potentially be used in any section (a “wild card”) if later processing suggests that that is beneficial.



```python
def score_dict(text, ref, d, headers):
    scores = {}
    ann = annotate_with_dict(text, d, headers, None, keep_overlaps=True)
    ann['start'] = ann['start'].astype(int)
    ann['end'] = ann['end'].astype(int)
    ann['concept_id'] = ann['concept_id'].astype(int)
    compare_ref_pred(ref.copy(), ann)
    scores_counter = Counter()
    for i in ann.index:
        h = ann['section'][i]
        source = ann['dict_entry'][i]
        k = (h, source)
        if k in d:
            if type(d[k]) == Counter:
                k = (k, ann['concept_id'][i][-1])
            scores.setdefault(k,[]).append(ann['score'][i])
            scores_counter[(k, ann['score'][i])] += 1
        else:
            print('key not in dict:', k)

    return scores, scores_counter
```

This function goes over the initial dictionary, that is constructed from the provided annotations, and for each dictionary key counts how many True Positive and False Positives it yields. These counts are then used to filter out keys which often lead to erroneous annotations. In particular, this filters out “wild card” entries which are so.



```
def limit_any_to_allowed_sections(d, allowed_sec, cid_to_type):
    any_keys = [k for k in d if k[0] == 'any']
    for k in any_keys:
        cid = d[k]
        if cid not in cid_to_type.index:
            print(f'CID {cid} not in flat snomed, skipping')
            continue
        ct = cid_to_type[cid]
        d[(tuple(allowed_sec[ct]), k[1])] = cid
        d.pop(k, None)
```

When expanding the dictionary with mentions which are the concept names in SNOMED CT, we do not have information about the sections in which they appear. However, we do know the type of concept – procedure, finding or body structure. By analyzing the statistics of which types appear in which section, we are able to limit to some extent the sections in which such new entries are allowed to be annotated.

6. Please provide the machine specs and time you used to run your model.
   - CPU (model): Intel(R) Core(TM) i7-10700 CPU @ 2.90GHz
   - GPU (model or N/A): N/A
   - Memory (GB): 128GB
   - OS: Windows 11
   - Train duration: 6 minutes
   - Inference duration: 5 documents/minute

7. Anything we should watch out for or be aware of in using your model (e.g. code quirks, memory requirements, numerical stability issues, etc.)?

8. Did you use any tools for data preparation or exploratory data analysis that aren't listed in your code submission?

   Some processing of the SNOMED CT data was used in an ad-hoc manner. The resultant files are included in the submission. Their creation process is as follows:
   We supplemented the dictionary with SNOMED-CT synonyms of the included concepts; and further expanded it with terms originating from other open vocabularies – e.g., ICD and READ – mapped to SNOMED-CT concepts by the Observational Health Data Sciences and Informatics (OHDSI) community (https://www.ohdsi.org/), as provided by the Athena open-source tool (https://athena.ohdsi.org/vocabulary/list).
   In the creation of the file of potential term extensions, we identified concept pairs, linked by the "Is A" relation, where the source name added a single word to the destination one (ignoring stop words).

   The acronyms dictionary is based on a list of medical abbreviations from https://github.com/imantsm/medical_abbreviations. Any abbreviation whose meaning is an exact match to a SNOMED-CT concept name or its synonym was mapped to that concept id.

9. How did you evaluate performance of the model other than the provided metric, if at all?

   The provided metric averages over concepts, which makes it strongly dependent on the number of documents in the evaluation (the more documents there are, the more rare concepts affect the score). As an alternative, we also considered a score which is based on documents. For a given document, consider the set of concepts which appear in it according to the gold standard and according to the prediction. The score of the document is the IoU over these sets of concepts. The final score is the (macro) average of these scores.

10. What are some other things you tried that didn't necessarily make it into the final workflow (quick overview)?

    We initially explored LLM-based solutions, and then LLM-based augmentations to the dictionary, but these seemed to work less well than a more conservative approach.
    An important part of the training is in filtering the annotated mentions. We tried using a more principled approach for doing so, based on a mathematical analysis of how the score changes as more mentions are detected for a given concept. But this worked less well than our initial ad-hoc heuristic.
    We tried some naïve linguistic rules for expanding the dictionary, some of which did not work well, and were discarded.

11. If you were to continue working on this problem for the next year, what methods or techniques might you try in order to build on your work so far? Are there other fields or features you felt would have been very helpful to have?

    We abandoned the LLMs approach because it became apparent that getting them to work well would require far more time than we had left. But if we had a year to work on this, we would revisit this.
    The annotations themselves seemed, at times, to be inconsistent and in some cases even wrong. Improving and expanding the annotations is likely to improve the models, especially in an LLM-based approach. This can be done in an iterative approach where annotators do not need to annotate documents "de-novo", but instead review the predicted annotations.

12. What simplifications could be made to run your solution faster without sacrificing significant accuracy?

    The solution is fairly simple. Run time could be improved by reducing the size of the dictionary. This can be done by offline filtering it versus a large corpus of relevant clinical notes. Dictionary keys which are never found in this corpus can be filtered out.