Building a Clinically-Focused Problem List From Medical Notes

Amir Feder* Itay Laish Shashank Agarwal Uri Lerner
Aviel Atias Cathy Cheung Peter Clardy Alon Cohen
Rachana Fellinger Hengrui Liu Lan Huong Nguyen Birju Patel
Natan Potikha Amir Taubenfeld Liwen Xu Seung Doo Yang
Ayelet Benjamini Avinatan Hassidim

Abstract

Clinical notes often contain useful information not documented in structured data, but their unstructured nature can lead to critical patient-related information being missed. To increase the likelihood that this valuable information is utilized for patient care, algorithms that summarize notes into a problem list have been proposed. Focused on identifying medically-relevant entities in the free-form text, these solutions are often detached from a canonical ontology and do not allow downstream use of the detected text-spans. Mitigating these issues, we present here a system for generating a canonical problem list from medical notes, consisting of two major stages. At the first stage, *annotation*, we use a transformer model to detect all clinical conditions which are mentioned in a single note. These clinical conditions are then grounded to a predefined ontology, and are linked to spans in the text. At the second stage, *summarization*, we develop a novel algorithm that aggregates over the set of clinical conditions detected on all of the patient's notes, and produce a concise patient summary that organizes their most important conditions.

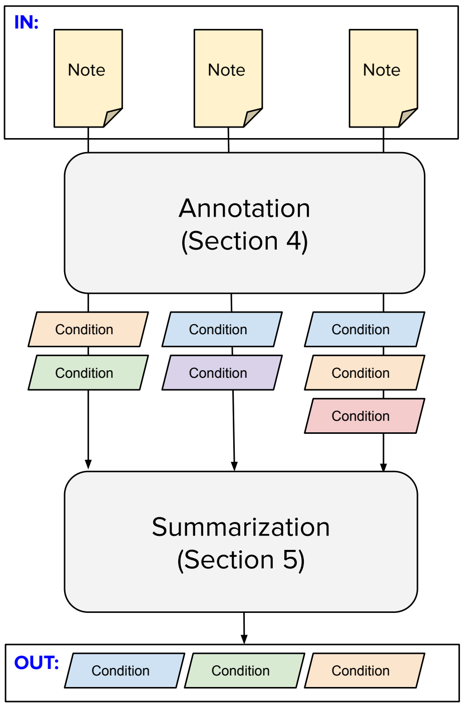

Figure 1: **System overview**: Conditions are extracted form each individual note at the *annotation* stage, and a single patient level list is generated from them at the *sumarization* stage.

1 Introduction

The pervasiveness of free-text narrative in Electronic Health Records (EHR) is both a blessing and a curse. It allows much more nuanced information about patients' conditions being saved and documented (Uzuner et al., 2010; Savova et al., 2010; Jensen et al., 2012; Wang et al., 2018; Feder et al., 2020). However, the unstructured nature of the data can also impede care givers' understanding of patient conditions (Walsh, 2004; Ford et al., 2016).

To allow care providers to better understand their patients' condition from medical notes, many machine learning (ML) models have been proposed (Uzuner et al., 2011; Jensen et al., 2012; Lee et al., 2020). These algorithms often solve a named-entity recognition (NER) task over the clinical notes, identifying text spans that correspond to clinical problems (Uzuner et al., 2011). While performance on such task has improved in the last decade (Wang et al., 2018), these models often do not link the identified entities to an ontology and are therefore sensitive to abbreviations, spelling errors and language ambiguity (Reátegui and Ratté, 2018; Gopinath et al., 2020; Gao et al., 2021). Moreover, these solutions operate at the note level, and are not able to aggregate a patient's overall medical problem list (Baumel et al., 2018). Both of these limitations decrease the utility of deploying these models in the real-world.

*Corresponding author (afeder@google.com).

Another important limitation of many existing solutions is that they are built on top of recurrent neural networks designed for solving NER tasks which often do not fully utilize the nuanced linguistic signal (Wang et al., 2018). These approaches were shown to produce very good results on de-identification tasks on clinical notes (Hartman et al., 2020), but can fail when presented with tasks that demand better understanding of context (Devlin et al., 2019). The recent transition of the entire NLP community to pre-trained transformer-based models (Wu et al., 2020) thus offers an opportunity to further improve on existing condition extraction methods (Zhu et al., 2018).

In this paper, we take on the task of addressing these problems and limitations, and describe how to build an end-to-end system that is robust and trustworthy. Concretely, given a set of notes describing a single patient, our goal is to output a clinically-focused problem list. Our system consists of two major stages: (1) *Annotation* (§4): operating at the level of a single medical note, we detect all clinical conditions which are mentioned in the text. These clinical conditions are then grounded to a predefined set of entities, and are linked to text spans. More formally, the output of the *annotation* stage is a set of tuples, where each tuple is a clinical condition identifier, a character span and context metadata (e.g., the acuity and presence of the condition). (2) *Summarization* (§5): operating at patient-level, we consume the set of clinical conditions detected during the *annotation* stage, and produce a concise patient summary that organizes the conditions. Our system is backed by a tailored *Ontology* (§3), defined on-top of SNOMED-CT (Donnelly et al., 2006) used by both stages to model the clinical knowledge required for this task. See Figure 1 for an illustration of our system.

2 Related Work

Identifying patient-related information in medical notes is long recognized as a core task in clinical-NLP. As such, there exist standardized datasets and competitions (Uzuner, 2009; Savova et al., 2010; Jensen et al., 2012; Ford et al., 2016; Zhu et al., 2018). The task of identifying medical concepts in clinical notes was organized as a competition in i2b2 2010 (Uzuner et al., 2011). In i2b2 and in subsequent work, this task was defined as a named entity recognition (NER) task (Hartman et al., 2020), where individual words are classified as to whether they contain medical problems. Subsequently, a Named Entity Normalization (NEN) task, where entities are standardized into known medical concepts, was later added to the i2b2 (now n2c2) competitions (Luo et al., 2019). Solutions to the problem consequently followed the conventional NLP approaches to solving NER tasks. Recent approaches harness the transformer architecture, solving a token-level binary classification task (Peng et al., 2019; Yadav and Bethard, 2019; Si et al., 2019; Lee et al., 2020).

To connect identified text spans to an ontology, a common solution is to look for the most similar entity in a given knowledge graph. Knowledge graphs use a graph-structured data model to integrate data (Ehrlinger and Wöß, 2016). They are often used to store interlinked descriptions of entities—objects, events, situations or abstract concepts—while also encoding the semantics underlying the used terminology. They were shown to be very useful in the medical domain and are often used to encode medical knowledge (Lindberg et al., 1993; Donnelly et al., 2006; Lipscomb, 2000). Specifically, in the context of free-form text, as that in the clinical notes, graph structured data models can be used to map many alternative descriptions of the same condition into one canonical definition (Organization, 2015).

Finally, the task of aggregating patient-related information across multiple documents into one problem list in a single system was not, to the best of our knowledge, published in prior research. The focus of our work is building an end-to-end system that connects the text *annotation* with the *summarization* stage.

3 Ontology

Our system is based on a universe of entities (*ontology*). The *ontology* captures the clinical knowledge required for our system to provide a concise and clinically-focused problem list. This knowledge improves both the detection of clinical conditions in medical notes (§4), and the subsequent bucketing of related conditions (§5).

On the detection side, it is necessary for our algorithm to be aware of the ways in which clinical conditions may appear in medical notes. For example, “iddm” (“insulin dependent diabetes mellitus”) is an alternative phrasing of “Diabetes mellitus type 1”, and “Miller” may refer to "Miller Fisher syndrome". On the bucketing side, it is necessary to have knowledge about related conditions (e.g., “Biventricular congestive heart failure” is related to “Right heart failure”) and about possible complications of certain conditions (e.g., “Diabetic nephropathy” is a complication of “Diabetes mellitus”).

Failure to capture this knowledge may increase the redundancy at the problem list level, and might cause dilution of signals and features, which in turn results in poor quality. The *ontology* is therefore a fundamental building block that is being used across all the system stages, and the way it is created has critical quality implications. In this section, we describe the creation of our *ontology*. Instead of creating a full *ontology* end-to-end, we have opted to base our *ontology* on pre-existing datasets. We collected a set of *Ground Truth Problem List*, which were curated by clinicians, and examined the properties of each dataset against this ground truth. A useful *ontology* should demonstrate the following properties:

(i) **High coverage** of the entries in the ground truth Problem List’s, and in the right granularity level.

(ii) **Easy to match** an entity from the ontology to the actual text in the medical note.

(iii) Entities should have **meaningful relationships** with other entities that are useful for reducing redundancy in the aggregated Problem List.

We considered multiple data sources, including: SNOMED-CT (Donnelly et al., 2006), MeSH (Lipscomb, 2000), ICD-10-CM (Organization, 2015), and UMLS (Lindberg et al., 1993).

3.1 ICD-10

.

ICD-10 is lacking some conditions (e.g., “Odynophagia”) violating property ((i)); a single main entity is missing for some conditions (e.g. “Sepsis” and “Pneumonia” are associated with multiple unrelated entities), these conditions are cluttered across the dataset, making it more difficult to group mentions together (violating property ((iii))); and due to the verbose description of some entities (e.g. “K44.9 Diaphragmatic hernia without obstruction or gangrene”), it is hard to match an ontology entity to the text (“Hiatal hernia” in the previous example), in violation of property ((ii)).

3.2 MeSH

In MeSH we observed some significant recall losses. For example, "Hypertensive urgency” and "Generalized anxiety disorder” were missing, violating property ((i)).

3.3 UMLS

Since UMLS is a combination of multiple systems, the relationships and granularity it provides vary across entities. This makes all properties only partially satisfied.

3.4 SNOMED-CT

While SNOMED-CT was missing some entities (e.g., "Right eye glaucoma”), these could usually be compensated by other SNOMED concepts without any significant clinical impact, and overall, it outperformed on all three properties the other options considered.

We note that due to the uniqueness in structure, relation types and granularity of each ontology, any attempt of reconciliation is exposed to similar issues as observed in *UMLS*. Therefore we chose to base our solution on a single ontology source (SNOMED), where each entity in our ontology corresponds to exactly one SNOMED concept. This allows us to maintain the consistency and granularity of SNOMED concepts and relationships, and also allows us to incorporate new versions of SNOMED as they are released, which keeps the ontology up to date.

Additionally, in order to enhance the ability to match a SNOMED concept to text from medical notes, we enrich SNOMED concepts with the followings (using our NameMapper algorithm described in section 4.2):

1. The ICD-10 codes of all ICD-10 diagnoses for which the SNOMED concept is their closest concept in SNOMED.
2. Phrases which are alternative ways to mention the entity in medical notes.

For (1), we use two sources for mapping ICD-10 diagnoses to their closest SNOMED concept: (a) OHDSI-OMOP (Stang et al., 2010; Hripcsak et al., 2015); and (b) the *NameMapper* (details in section 4.2) algorithm, applied on a diagnosis’ name in order to match it against the set of SNOMED terms. For (2), we consider clusters of phrases that are originated from various sources: MeSH, UMLS and manually-curated abbreviations. All phrases in a cluster refer to the same entity. We use NameMapper again, in order to match each phrase in a cluster against the full set of SNOMED terms. We add the entire cluster to the entity that corresponds to the closest SNOMED concept.

4 Annotation

The annotation stage is performed at the level of a single clinical note. At the end of this stage each mention of a condition in the text is exported in the form of $(ConditionID, (start, end), ContextInfo)$ tuple (where $start$, end refer to a char offset from the beginning of the note), $ConditionID$ is a unique entry in the ontology described in §3, and $ContextInfo$ includes extracted information about the condition, such as acuity, presence, etc.

We start this section by describing our detection (§4.1), and mapping (§4.2) algorithms.

4.1 ML model for surfacing candidates and context information

We extract condition spans from free text using an ML NER model. Later, we try to map these candidate spans into our ontology (§4.2).

Our model is a multi-task encoder-only transformer model (BERT; Devlin et al., 2019). Its main task is a 4-class classification task (using the labels **PROBLEM**, **BODY PART**, **QUALIFIER**, **PROCEDURE**), with additional two supplementary-tasks:

- Existences: For each of the four labels, whether it is PRESENT or ABSENT; e.g., in "ruled out cancer", "cancer" is labeled as *ABSENT*.
- Relation: For both BODY_PARTS / QUALIFIERS, are they associated with the PROBLEM / PROCEDURE on their left, or on their right. E.g., in "diabetic foot ulcer", "ulcer" will have a LEFT_HAND_SIDE label. This information is later used to map the annotated term to the most accurate ontology entity.

This is the 2nd generation of NER models used in our system, our previous model was based on GloVe, Bi-LSTM and CRF (Hartman et al., 2020). On top of the CRF layer we placed three softmax layers to solve each of the three aforementioned tasks (this model is referred as *Bi-LSTM* below).

The BERT model described here showed superior performance (see table below). For BERT, we use a similar approach where we place three softmax layers on top of the pre-trained contextual embedding. The added layers are then fine-tuned on the MIMIC-III dataset (Johnson et al., 2016; using the same labels of the Bi-LSTM). We experimented with 3 pre-trained BERTs:

BERT-base from the original paper (Devlin et al., 2019).

BERT-small based on (Turc et al., 2019) – x2 more efficient than *BERT-base*.

PubMED BERT same architecture as *BERT-base*, pre-trained from scratch on MEDLINE/PubMed, using the original uncased word-piece tokenizer (Lee et al., 2020).

The labels are split $80\%/20\%$ for train/eval sets. The following table shows the results on the eval set. As can be seen, *PubMED BERT* surpasses the other models.

4.2 NameMapper – A graph traversal-based approach for ontology matching

In many cases, the hand-written text by clinicians in notes does not match the names of conditions in the ontology. To bridge this gap, and to increase the coverage of problems detected and matched by our algorithm in §4.1, we introduce a graph-based fuzzy text matcher called *NameMapper*. The NameMapper is used during the following stages of our system:

(i) **Ontology creation (§3)**: For mapping between entities in different ontologies (ICD-10→ SNOMED).
(ii) **Increase detection coverage**: Build a vocabulary used at the annotation stage for matching text spans to entities from the ontology.
(iii) **Mapping**: Map conditions spans generated in §4.1 to entities in our ontology.

NameMapper is essentially a string matching algorithm. It operates on text that is suspected to match a name of an entity in the ontology. It expands the string using different variations of each word within, and allows string manipulations using pre-defined operations. Each operation is associated with a cost. We use a graph (with these costs set as edge weights) to find the closest entity in the ontology to the input string. See an illustration in Figure 3). The process consists of three main stages:

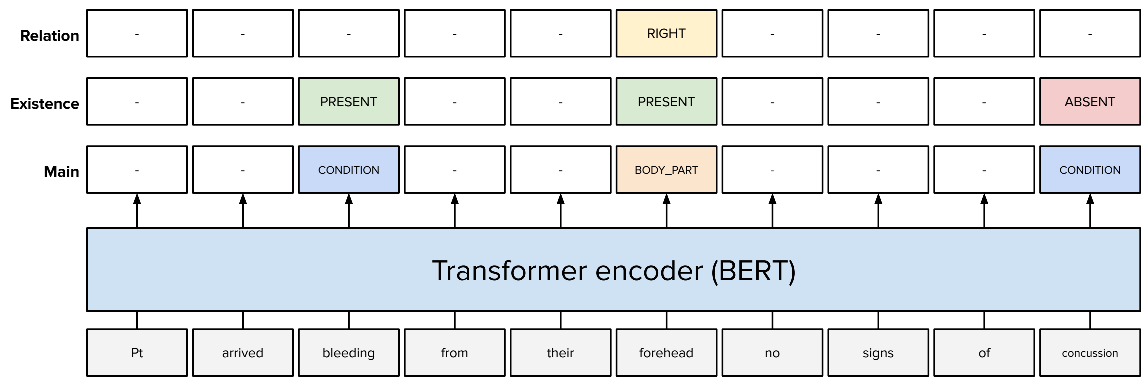

Figure 2: **Illustration of our multi-task encoder-only transformer**: Each token is labeled for *type classification(Main)*, *Existence* and *Relation*.

Parsing. We first break the input string into a (non-intersecting) sequence of name components of different types: e.g., connectors such as "due to" or "of" are modeled as a special type.

Generation. For each name component, we generate a set of alternatives, each alternative is associated with a cost. These alternatives represent different ways to refer to the same concept, e.g., "malignant tumor" → "cancer", "lung" → "pulmonary", "kidney" → "renal", "diabetes mellitus" → "diabetes"). These costs were manually curated. One could think of them as the conceptual distance between the two synonyms (e.g., replacing "ii" with "2" has a lower cost compared to replacing "infectious disease" with "infection"). The alternating names include the original string as it appears in the input name (up to lower-casing and some other default operations) and alternative wordings that are based on synonyms, dropping optional phrases, stemming and more. We manually curated those rules. For example, *"diabetes"* is a synonym of *diabetes mellitus*, *diabet* is the canonical form of *"diabetes"*.

Using the alternatives of each name component, we generate a list of alternative writings for the entire phrase. The alternatives include different combinations of the options created for the name components generated during the previous stage. This stage also allows phrase level transformations (with additional cost). For example, the connector "due to" allows a transformation of dropping itself and the possibility of swapping both of its sides: "coma due to diabetes mellitus" may generate alternatives such as "coma diabetes mellitus" and "diabetes mellitus coma" (each with a cost). The final (phrase level) cost is set to be the costs sum of all replacements and operations applied to the input string.

Selection. We output the ontology entity that matches the best candidate (lowest cost). For example, the terms "diabetic coma", "coma due to diabetes mellitus" and "diabetes mellitus coma" will all be mapped to the same ontology entity *"Coma due to diabetes mellitus"*, each with a cost.

5 Summarization

The *Annotation* Phase (§4) outputs the mentions of clinical conditions in the medical notes. The goal of the *Summarization* Phase is to take all the mentions across all the notes and generate a comprehensive and coherent problem list, optimized for the needs of clinicians who care for the patient. In addition to the mentions themselves, the Summarization Phase can use additional information in the patient's chart

	PROBLEM [3.8K]			BODY PART [1.4K]			QUALIFIER [0.7K]			PROCEDURE [0.4K]		
	P	R	F1	P	R	F1	P	R	F1	P	R	F1
Bi-LSTM	84.74	86.80	85.76	**75.52**	80.02	77.70	60.65	55.17	57.78	**71.83**	64.15	67.77
PubMED BERT	**87.64**	**89.69**	**88.65**	72.61	**86.79**	**79.07**	**65.27**	**62.18**	**63.69**	65.94	**76.58**	**70.86**
BERT base	86.47	87.19	86.83	74.22	82.51	78.15	61.24	57.77	59.45	64.99	67.80	66.36
BERT small	84.10	86.65	85.35	69.82	81.22	75.09	59.51	55.83	57.61	62.13	57.36	59.65

Table 1: ML model classification results

as well as general medical knowledge. We now describe the sequence of steps that make up the Summarization Phase.

5.1 Grouping

The first step is to collect all the clinical condition mentions related to the same condition. In this step, we drop conditions that the patient never had (e.g., mentions of known side effects of treatments, speculations written in the note etc.) using the existence signal generated by our annotator (§4.1).

5.2 Bucketing

In the next step, we group clinical conditions that are related to each other. For example, if we found mentions of *Systolic Heart Failure*, *Diastolic Heart Failure*, *Acute Heart Failure*, and *Acute Diastolic Heart Failure* in a patient's medical notes, we would bucket those mentions under a *Heart Failure* bucket, even if "Heart Failure" itself was not explicitly mentioned in the patient's record.

In the example above, *Heart Failure* is an anchor entity, used to bucket together more specific conditions as defined by the is-a relation of the SNOMED ontology (see Section 3). A bucket is defined as a collection of patient conditions composed of one or more anchor entities and their corresponding descendants (in the ontology).

Ideally, conditions inside a bucket should involve similar pathophysiologies, medications and therapies. Anchor entities should thus follow the Goldilocks Principle and be neither too broad nor too narrow. Overly broad anchor entities (e.g., *Heart disease*) represent conditions with very different pathophysiologies and therapies and therefore do not provide a good clinical view. Overly narrow anchor entities (e.g., *Systolic Heart Failure*, *Diastolic Heart Failure*, *Acute Heart Failure*, and *Acute Diastolic Heart Failure*) would make the Problem List overly long and redundant, reduc-

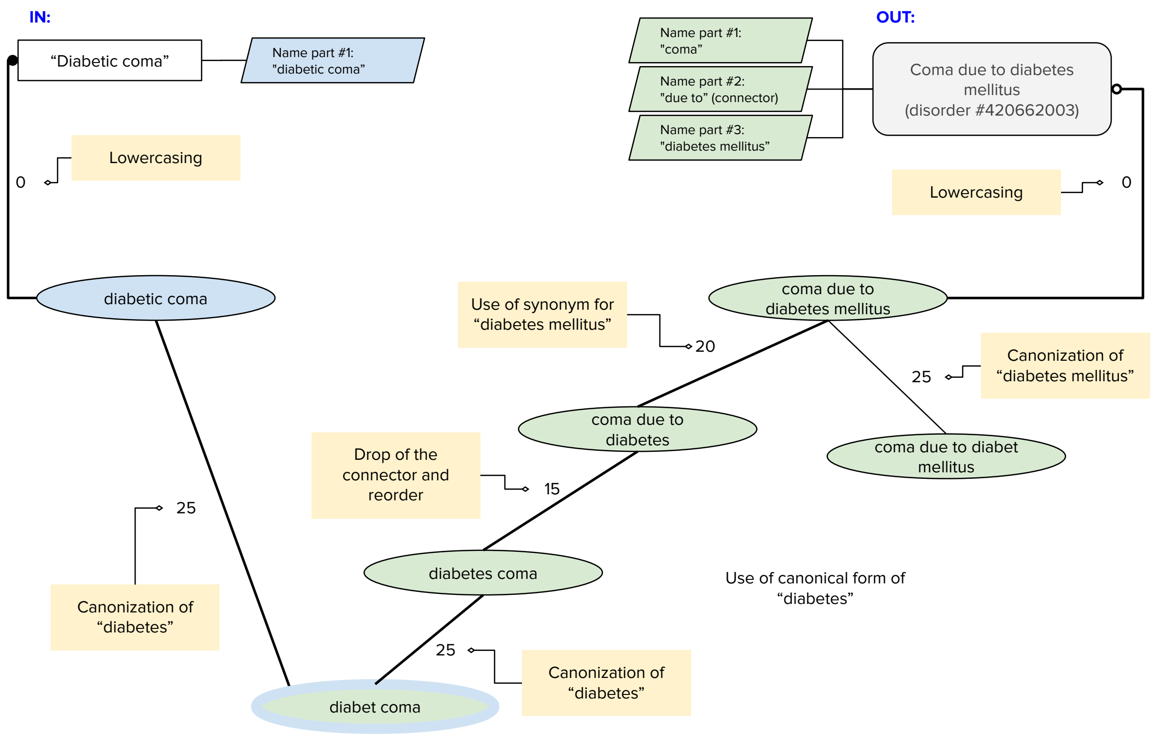
IN:
"Diabetic coma"
Name part #1: "diabetic coma"
Lowercasing
0
diabetic coma
25
Canonization of "diabetes"
diabet coma
25
Canonization of "diabetes"
diabetes coma
Drop of the connector and reorder
15
coma due to diabetes
Use of synonym for "diabetes mellitus"
20
coma due to diabetes mellitus
25
Canonization of "diabetes mellitus"
coma due to diabet mellitus
Use of canonical form of "diabetes"
Name part #1: "coma"
Name part #2: "due to" (connector)
Name part #3: "diabetes mellitus"
OUT:
Coma due to diabetes mellitus (disorder #420662003)
Lowercasing
0

Figure 3: **Illustration of the NameMapper**: At stage **Parsing** the input *"Diabetic coma"* is parsed to its single name component (blue parallelogram). Additionally, the ontology entity *"Coma due to diabetes mellitus"* is parsed to *Coma*, *diabetes mellitus* and the connector *due to* (green parallelograms). Then the generation stage **Generation** will use the variations of each name component to create all possible permutations both for the input and the ontology entity(ies), given in blue and green ovals accordingly. Finally stage **Selection** will find the shortest path between the input and the ontology, that is *Coma due to diabetes mellitus* with the cost of 85 (in bold).

ing its usefulness. Identifying good anchor entities requires clinical expertise.

As part of the Bucketing step, we also determine the name of the bucket. This is typically the name of the anchor entity or entities around which the bucket is defined. However, if the bucket only contains more specific entities than the anchor entities, we give the bucket a more specific name.

For example, if the only conditions in the bucket anchored on *Heart Failure* were *Diastolic Heart Failure*, and *Acute Diastolic Heart Failure*, instead of naming the bucket *"Heart Failure"*, we would name it *"Diastolic Heart Failure"*, which is a more accurate description of all the entities that ended up in the bucket for this particular patient.

5.2.1 Secondary buckets

Some conditions are associated with other conditions typically but not always. For example, *Hyperglycemia* is often associated with *Diabetes Mellitus* but it is possible to have *nondiabetic Hyperglycemia*. In this case we consider *Diabetes Mellitus* as a secondary bucket of *Hyperglycemia*, meaning that if the *Diabetes Mellitus* bucket includes other conditions then it should also include *Hyperglycemia*, but if it is empty (does not exist) then *Hyperglycemia* should be its own bucket.

5.3 Bucket Presence

At the end of the bucketing step, we generate collections of clinical conditions that were mentioned in the patient's medical notes. However, the patient does not necessarily have all conditions that were mentioned. The typical reasons are mistakes in the Annotation Phase or actual uncertainty, e.g., a patient may have a mention indicating that Covid-19 is likely only to be ruled out in a later mention. Obviously, it is desirable to omit these conditions from the list, as the patient does not have them.

In the Condition Bucket Presence step, we determine if the patient is having, or has ever had each of the condition buckets. First, we make use of the Existence signal extracted during the Annotation phase (see Section 4.1), and drop the mentions classified as "ABSENT" by the algorithm based on the surrounding context. Since (as expected) the Annotator's existence classification is not always perfect, we apply an additional second level to improve the presence detection. To handle mistakes from the Annotation Phase we take into account the frequency with which the condition was mentioned in medical notes, the section where the condition was mentioned, and the credentials of the medical note's author. We are looking into using additional signals such as mentions in the notes of related conditions, documentations of conditions in the EHR structured Problem Lists, information about labs, vitals and medications, and many others.

5.4 Classification

Over time, a patient is expected to have many clinical conditions. However, not all conditions are active when the patient is reviewed, and if they all were to be displayed, the list would quickly become overwhelmingly long and not particularly useful. Imagine reading a chart of a patient that caught a common cold a few years ago who is also diabetic; more details about diabetes and related conditions should be surfaced, and the information about the common cold's occurrence in the far past would be no longer relevant today, and therefore should be skipped to avoid unnecessary clutter. Since our Annotation phase also detects symptoms and procedures seen in the patient's medical notes, the length of the generated Problem List can become extremely long. It is thus important for us to strive for conciseness, and avoid information overload that could distract physicians from the important active conditions.

To make the Problem List easier to comprehend, we classify the clinical conditions into four categories: *Active Conditions*, *Historical Conditions*, *Procedures*, and *Symptoms*.

SNOMED classifies every entity into a type which includes disorders, findings, and procedures. We consider the SNOMED types of all the entities in a bucket to determine the type of the bucket: a bucket with disorders is a Conditions bucket, a bucket with findings is a Symptoms bucket and so on. We have additional logic for mixed buckets, e.g., a mixture of disorders and findings is considered a Conditions bucket.

In the next step we classify the Conditions buckets into Active and Historical category. We do this by first classifying individual conditions included in a bucket separately and then again classifying the entire bucket based on the whole collection: a bucket with at least one Active condition is Active, otherwise it is Historical. A chronic lifelong condition such as *Type 1 Diabetes Mellitus* is always considered *Active*. The remaining conditions are considered *Active* and then moved to *Historical* if a mention confirming their presence wasn't seen for a long time, or if a mention was found explicitly indicating that the condition was resolved. The duration after which a non-lifetime condition is automatically classified as *Historical* (because it was not mentioned again as present) varies, and is part of our curated knowledge gathered with assistance of expert clinicians.

5.5 Ranking

Finally, we rank the conditions in each category so that the most clinically important conditions are displayed first. Our ranking function accounts for the severity and recency clinical conditions to determine the order. More severe and more recent conditions are ranked higher to highlight the conditions that might require more attention from physicians.

5.6 Summarization evaluation

Each step in the *Summarization* Phase is evaluated separately so that we are able to test those steps in isolation. At the same time, we also test the overall pipeline by evaluating the resulting Problem List holistically. In addition to evaluating metrics such as precision and recall, we also measure the usefulness of the Problem List, which captures the effects of steps such as Bucketing, Classification, and Ranking.

6 Conclusion

In this work, we present an end-to-end system for summarizing a patient's problem list directly from their entire collection of medical notes. This system aggregates over identified conditions in each note, producing a concise list mapped to a canonical ontology and without duplicated conditions. Building on recent improvements in natural language understanding models, especially encoder-only transformers, we show how NLP models can be used as part of an holistic system. We hope that our work will spur more research on how to utilize NLP for better, more robust and trustworthy, health informatics systems.

References

Tal Baumel, Jumana Nassour-Kassis, Raphael Cohen, Michael Elhadad, and Noémie Elhadad. 2018. Multi-label classification of patient notes: case study on icd code assignment. In *Workshops at the thirty-second AAAI conference on artificial intelligence*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186.

Kevin Donnelly et al. 2006. Snomed-ct: The advanced terminology and coding system for ehealth. *Studies in health technology and informatics*, 121:279.

Lisa Ehrlinger and Wolfram Wöß. 2016. Towards a definition of knowledge graphs. *SEMANTiCS (Posters, Demos, SuCCESS)*, 48(1-4):2.

Amir Feder, Danny Vainstein, Roni Rosenfeld, Tzvika Hartman, Avinatan Hassidim, and Yossi Matias. 2020. Active deep learning to detect demographic traits in free-form clinical notes. *Journal of Biomedical Informatics*, 107:103436.

Elizabeth Ford, John A Carroll, Helen E Smith, Donia Scott, and Jackie A Cassell. 2016. Extracting information from the text of electronic medical records to improve case detection: a systematic review. *Journal of the American Medical Informatics Association*, 23(5):1007–1015.

Yanjun Gao, Dmitriy Dligach, Leslie Christensen, Samuel Tesch, Ryan Laffin, Dongfang Xu, Timothy Miller, Ozlem Uzuner, Matthew M Churpek, and Majid Afshar. 2021. A scoping review of publicly available language tasks in clinical natural language processing. *arXiv preprint arXiv:2112.05780*.

Divya Gopinath, Monica Agrawal, Luke Murray, Steven Horng, David Karger, and David Sontag. 2020. Fast, structured clinical documentation via contextual autocomplete. In *Machine Learning for Healthcare Conference*, pages 842–870. PMLR.

Tzvika Hartman, Michael D Howell, Jeff Dean, Shlomo Hoory, Ronit Slyper, Itay Laish, Oren Gilon, Danny Vainstein, Greg Corrado, Katherine Chou, et al. 2020. Customization scenarios for de-identification of clinical notes. *BMC medical informatics and decision making*, 20(1):1–9.

George Hripcsak, Jon D Duke, Nigam H Shah, Christian G Reich, Vojtech Huser, Martijn J Schuemie, Marc A Suchard, Rae Woong Park, Ian Chi Kei Wong, Peter R Rijnbeek, et al. 2015. Observational health data sciences and informatics (ohdsi): opportunities for observational researchers. *Studies in health technology and informatics*, 216:574.

Peter B Jensen, Lars J Jensen, and Søren Brunak. 2012. Mining electronic health records: towards better research applications and clinical care. *Nature Reviews Genetics*, 13(6):395–405.

Alistair EW Johnson, Tom J Pollard, Lu Shen, Li-wei H Lehman, Mengling Feng, Mohammad Ghassemi, Benjamin Moody, Peter Szolovits, Leo Anthony Celi, and Roger G Mark. 2016. Mimic-iii, a freely accessible critical care database. *Scientific data*, 3(1):1–9.

Jinhyuk Lee, Wonjin Yoon, Sungdong Kim, Donghyeon Kim, Sunkyu Kim, Chan Ho So, and Jaewoo Kang. 2020. Biobert: a pre-trained biomedical language representation model for biomedical text mining. *Bioinformatics*, 36(4):1234–1240.

Donald AB Lindberg, Betsy L Humphreys, and Alexa T McCray. 1993. The unified medical language system. *Yearbook of medical informatics*, 2(01):41–51.

Carolyn E Lipscomb. 2000. Medical subject headings (mesh). *Bulletin of the Medical Library Association*, 88(3):265.

Yen-Fu Luo, Weiyi Sun, and Anna Rumshisky. 2019. Mcn: a comprehensive corpus for medical concept normalization. *Journal of biomedical informatics*, 92:103132.

World Health Organization. 2015. International classification of diseases, tenth revision, (icd-10).

Yifan Peng, Shankai Yan, and Zhiyong Lu. 2019. Transfer learning in biomedical natural language processing: an evaluation of bert and elmo on ten benchmarking datasets. *arXiv preprint arXiv:1906.05474*.

Ruth Reátegui and Sylvie Ratté. 2018. Comparison of metamap and ctakes for entity extraction in clinical notes. *BMC medical informatics and decision making*, 18(3):13–19.

Guergana K Savova, James J Masanz, Philip V Ogren, Jiaping Zheng, Sunghwan Sohn, Karin C Kipper-Schuler, and Christopher G Chute. 2010. Mayo clinical text analysis and knowledge extraction system (ctakes): architecture, component evaluation and applications. *Journal of the American Medical Informatics Association*, 17(5):507–513.

Yuqi Si, Jingqi Wang, Hua Xu, and Kirk Roberts. 2019. Enhancing clinical concept extraction with contextual embeddings. *Journal of the American Medical Informatics Association*, 26(11):1297–1304.

Paul E Stang, Patrick B Ryan, Judith A Racoosin, J Marc Overhage, Abraham G Hartzema, Christian Reich, Emily Welebob, Thomas Scarnecchia, and Janet Woodcock. 2010. Advancing the science for active surveillance: rationale and design for the observational medical outcomes partnership. *Annals of internal medicine*, 153(9):600–606.

Iulia Turc, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. Well-read students learn better: On the importance of pre-training compact models. *arXiv preprint arXiv:1908.08962*.

Özlem Uzuner. 2009. Recognizing obesity and comorbidities in sparse data. *Journal of the American Medical Informatics Association*, 16(4):561–570.

Özlem Uzuner, Imre Solti, and Eithon Cadag. 2010. Extracting medication information from clinical text. *Journal of the American Medical Informatics Association*, 17(5):514–518.

Özlem Uzuner, Brett R South, Shuying Shen, and Scott L DuVall. 2011. 2010 i2b2/va challenge on concepts, assertions, and relations in clinical text. *Journal of the American Medical Informatics Association*, 18(5):552–556.

Stephen H Walsh. 2004. The clinician's perspective on electronic health records and how they can affect patient care. *Bmj*, 328(7449):1184–1187.

Yanshan Wang, Liwei Wang, Majid Rastegar-Mojarad, Sungrim Moon, Feichen Shen, Naveed Afzal, Sijia Liu, Yuqun Zeng, Saeed Mehrabi, Sunghwan Sohn, et al. 2018. Clinical information extraction applications: a literature review. *Journal of biomedical informatics*, 77:34–49.

Stephen Wu, Kirk Roberts, Surabhi Datta, Jingcheng Du, Zongcheng Ji, Yuqi Si, Sarvesh Soni, Qiong Wang, Qiang Wei, Yang Xiang, et al. 2020. Deep learning in clinical natural language processing: a methodical review. *Journal of the American Medical Informatics Association*, 27(3):457–470.

Vikas Yadav and Steven Bethard. 2019. A survey on recent advances in named entity recognition from deep learning models. *arXiv preprint arXiv:1910.11470*.

Henghui Zhu, Ioannis Ch Paschalidis, and Amir Tahmasebi. 2018. Clinical concept extraction with contextual word embedding. *arXiv preprint arXiv:1810.10566*.