Journal of the American Medical Informatics Association, 27(10), 2020, 1510–1519
doi: 10.1093/jamia/ocaa080
Advance Access Publication Date: 27 July 2020
Research and Applications

Research and Applications

Unified Medical Language System resources improve sieve-based generation and Bidirectional Encoder Representations from Transformers (BERT)–based ranking for concept normalization

Dongfang Xu [1], Manoj Gopale [2], Jiacheng Zhang [3], Kris Brown [4], Edmon Begoli [4] and Steven Bethard [1]

[1]School of Information, University of Arizona, Tucson, Arizona, USA, [2]Department of Electrical and Computer Engineering, University of Arizona, Tucson, Arizona, USA, [3]Department of Computer Science, University of Arizona, Tucson, Arizona, USA, and [4]National Center for Computational Sciences, Oak Ridge National Laboratory, Oak Ridge, Tennessee, USA

Corresponding Author: Dongfang Xu, School of Information, University of Arizona, 1103 E. 2nd St., Harvill Building, Rm 437D, Tucson, AZ 85721, USA; dongfangxu9@email.arizona.edu

Received 9 February 2020; Revised 25 March 2020; Editorial Decision 19 April 2020; Accepted 27 April 2020

ABSTRACT

Objective: Concept normalization, the task of linking phrases in text to concepts in an ontology, is useful for many downstream tasks including relation extraction, information retrieval, etc. We present a generate-and-rank concept normalization system based on our participation in the 2019 National NLP Clinical Challenges Shared Task Track 3 Concept Normalization.
Materials and Methods: The shared task provided 13 609 concept mentions drawn from 100 discharge summaries. We first design a sieve-based system that uses Lucene indices over the training data, Unified Medical Language System (UMLS) preferred terms, and UMLS synonyms to generate a list of possible concepts for each mention. We then design a listwise classifier based on the BERT (Bidirectional Encoder Representations from Transformers) neural network to rank the candidate concepts, integrating UMLS semantic types through a regularizer.
Results: Our generate-and-rank system was third of 33 in the competition, outperforming the candidate generator alone (81.66% vs 79.44%) and the previous state of the art (76.35%). During postevaluation, the model's accuracy was increased to 83.56% via improvements to how training data are generated from UMLS and incorporation of our UMLS semantic type regularizer.
Discussion: Analysis of the model shows that prioritizing UMLS preferred terms yields better performance, that the UMLS semantic type regularizer results in qualitatively better concept predictions, and that the model performs well even on concepts not seen during training.
Conclusions: Our generate-and-rank framework for UMLS concept normalization integrates key UMLS features like preferred terms and semantic types with a neural network–based ranking model to accurately link phrases in text to UMLS concepts.

Key words: deep learning, unified medical language system, natural language processing, concept normalization, generate-and-rank

INTRODUCTION

Background and Significance

Mining and analyzing the constantly growing unstructured text in the biomedical domain offers great opportunities to advance scientific discovery[1,2] and improve the clinical care.[3,4] However, lexical and grammatical variations are pervasive in such text, posing key challenges for data interoperability and the development of natural language processing (NLP) techniques.

For instance, *cardiovascular stroke*, *heart attack*, and *MI* all refer to the same concept. It is critical to disambiguate these terms by linking them with their corresponding concepts in an ontology. Such linking allows downstream tasks (relation extraction, information retrieval, etc.) to access the ontology's rich knowledge about biomedical entities, their synonyms, semantic types, and mutual relationships.

Concept normalization (CN) is a task that maps concept mentions, the in-text natural-language mentions of ontological concepts, to concept entries in an ontology. As one of the most comprehensive biomedical ontologies, the Unified Medical Language System (UMLS)[5] has been widely used in CN such as clinical disorder normalization in 2013 ShARe/CLEF eHealth[6] and 2014 SemEval Task 7 Analysis of Clinical Text,[7] and adverse drug reaction normalization in Social Media Mining for Health shared tasks.[8,9] The organizers of the 2019 n2c2 (National NLP Clinical Challenges) shared task track 3 Clinical Concept Normalization adopted the medical concept normalization (MCN) corpus,[10] which is annotated with a broader set of medical concepts than the disease or disorder concepts of previous work. The work described in the current article is based on our participation in the 2019 n2c2 shared task.

Traditional approaches for CN mainly include dictionary lookup. These approaches differ in how they construct dictionaries, such as collecting concept mentions from the labeled data as extra synonyms,[11,12] and what string-matching techniques are used, such as string edit distance.[13] Two of the most commonly used tools, MetaMap[14] and cTAKES (clinical Text Analysis Knowledge Extraction System),[15] both employ rules to first generate lexical variants for each noun phrase and then conduct dictionary lookup for each variant. However, end users have to make substantial efforts to achieve reasonable performance by customizing different modules.[16,17] A few recent CN systems[18,19] have demonstrated that rule-based approaches achieve performance competitive with other approaches in a sieve-based approach that carefully selects combinations and orders of dictionaries, exact and partial matching, and heuristic rules. However, such rule-based approaches struggle when there are great variations between concept mention and concept.

On the one hand, owing to the availability of shared tasks and annotated data, the field has shifted toward supervised machine learning techniques. Such approaches can be divided into 2 categories, classification[20–28] and learning to rank (LtR).[29–33] Classification-based approaches using deep neural networks have shown strong performance. They consider various architectures, such as gated recurrent units with attention mechanisms[26] and multitask learning with auxiliary tasks to generate attention weights[27]; sources for training word embeddings, such as Google News[22] or concept definitions from the UMLS Metathesaurus[24]; and input representations, such as character embeddings.[27] All classification approaches share the disadvantage that the size of the output space is the number of concepts to be predicted, so these approaches typically consider only a limited number of concepts, for example, 2200 concepts in Limsopatham et al[22] and around 22 500 concepts in Weissenbacher et al.[8] Classification approaches cannot predict concepts that were not seen in the training data.

On the other hand, pointwise LtR,[29,31] pairwise LtR,[30,32] and listwise LtR[33] approaches to CN can handle ontologies with millions of concepts, as they first reduce the output space to a small list of candidate concepts, and then rank the concepts in that list. DNorm,[30] which applied pairwise LtR in which mentions and concept names were represented as TF-IDF vectors, was the first to use LtR for CN and achieved the best performance in the ShARe/CLEF eHealth 2013 shared task. Listwise LtR approaches are both computationally more efficient than pairwise LtR[34] and empirically outperform both pointwise and pairwise approaches.[35]

Pretrained language models such as ELMo (Embeddings from Language Models)[36] and Bidirectional Encoder Representations from Transformers (BERT)[37] have shown great improvements in NLP tasks ranging from sentiment analysis to named entity recognition to question answering. Recently, BERT has been adapted to the biomedical domain by further pretraining on corpora such as PubMed abstracts and PubMed Central full-text articles (BioBERT),[38] clinical notes in the MIMIC-III (Medical Information Mart for Intensive Care) (Clinical-BERT),[39–41] or combinations of both sources.[39,42] Such biomedical domain specific BERTs achieve great performance on various biomedical NLP tasks, including concept extraction,[39,41,42] concept normalization,[43–45] relation extraction,[38,42,46,47] readmission prediction,[40] natural language inference,[42] etc. Among all the BERT-based CN systems, 2 systems take classification-based approaches,[43,45] with one showing that a BERT-based classifier performs poorly when the concept space is large (around 380 000 concepts in the ontology)[43] and the other[44] taking a generate-and-rank approach similar to ours, but they do not leverage resources such as synonyms or semantic type information from UMLS in their BERT-based ranker.

OBJECTIVE

We propose an architecture (shown in Figure 1) that is able to consider both morphological and semantic information. We first design a candidate generator (component [**a-e**]) to generate a list of candidate concepts, and then use a listwise classifier based on the BERT neural network[37] to rank them. In contrast to previous listwise classifiers,[33] which only take the concept mention as input, our BERT-based listwise classifier takes both the concept mention and the candidate concept name as input, and is thus able to handle concepts that never appear in the training data. We further enhance this listwise approach with a semantic type regularizer (ST) that allows our ranker to leverage UMLS semantic type information during training. Our proposed CN framework achieved the third-highest performance at the 2019 n2c2 shared task track 3.

MATERIALS AND METHODS

Materials

Dataset

All experiments were performed on the MCN corpus, which consists of 13 609 concept mentions drawn from 100 discharge summaries from the fourth i2b2/VA shared task.[48] The mentions are mapped to 3792 unique concepts of 434 056 possible concepts with 125 semantic types in SNOMED-CT (Systematized Nomenclature of Medicine Clinical Terms) and RxNorm. We take 40 discharge summaries from the released data as the training set, and the remaining 10 as the dev set, and keep the standard evaluation as the test set. We summarize dataset characteristics in Table 1. Few mentions are ambiguous in the

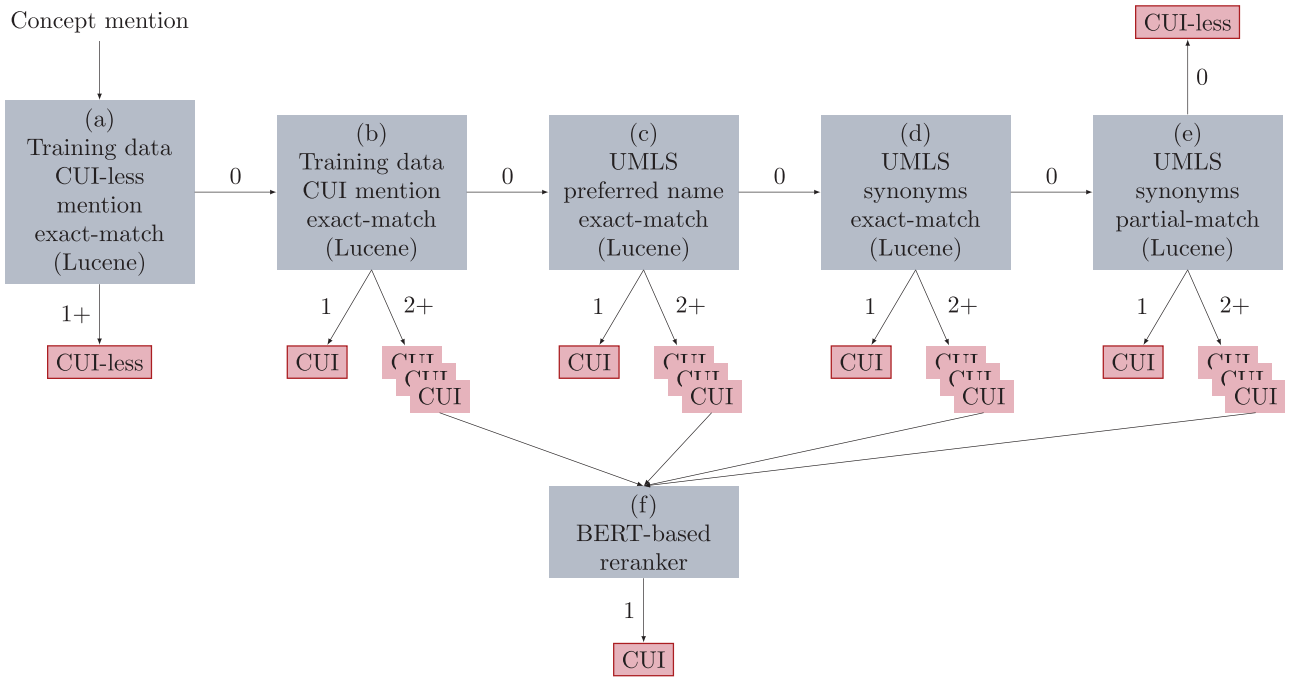

Figure 1. Architecture of our proposed framework for concept normalization. The edges out of a search process indicate the number of matches necessary to follow the edge. Outlined nodes are terminal states that represent the predictions of the model. Candidate generator (a–e). Candidate reranker (f). BERT: Bidirectional Encoder Representations from Transformers; CUI: concept unique identifier; UMLS: Unified Medical Language System.

Table 1. Dataset statistics of the medical concept normalization corpus

Statistics	Train	Dev	Test
# of documents	40	10	50
# of mentions	5334	1350	6925
# of unique concepts	1981	755	2579
$\frac{\text{of unseen mentions}}{\text{of mentions}}$ (%)	—	53.48	50.76
$\frac{\text{of unseen concepts}}{\text{of mentions}}$ (%)	—	32.37	29.85
$\frac{\text{of CUI-less}}{\text{of mentions}}$ (%)	2.32	2.00	3.13
$\frac{\text{of mentions}}{\text{of unique concepts}}$	2.69	1.79	2.69
$\frac{\text{of ambiguous mentions}}{\text{of mentions}}$ (%)	3.30	1.04	2.77

The # of unseen mentions for dev indicates the # of mentions that do not appear in the training set but do appear in the dev set. The # of unseen mentions for test indicates # of mentions that do not appear in the training or dev set but do appear in the test set. The # of unseen concepts for dev indicates the # of mentions whose normalized concepts do not appear in the training set but do appear in the dev set. The # of unseen concepts for test indicates the # of mentions whose normalized concepts do not appear in the training or dev set but do appear in the test set. The # of CUI-less indicates the # of mentions that could not be mapped to any concepts in the ontology. The # of ambiguous mentions indicates the # of mentions that could be mapped to more than 1 concept in the dataset.
CUI: concept unique identifier.

data—around 1% in the dev set and 2.8% in the test set—so the context in which mentions appear plays only a minor role in these data. A major challenge is the unseen mentions and concepts: 50.76% (29.85%) of test mentions (concepts) were not seen in the training or dev data. Systems that memorize the training data or rely on it to determine the space of output concepts will thus perform poorly.

Unified Medical Language System

The UMLS Metathesaurus[5] links similar names for the same concept from nearly 200 different vocabularies including SNOMED-CT, RxNorm, etc. There are over 3.5 million concepts in UMLS, and for each concept, UMLS also provides the definition, preferred term, synonyms, semantic type, relationships with other concepts, etc. Following the procedure in Luo et al,[10] we restrict our concepts to the 2 vocabularies of SNOMED-CT and RxNorm in UMLS version 2017AB. In our experiments, we make use of UMLS preferred terms, synonyms, and semantic types of these concepts. Synonyms (English only) are collected from level 0 terminologies containing vocabulary sources for which no additional license agreements are necessary.

System description

System architecture

We define a concept mention m as an abbreviation such as *MI*, a noun phrase such as *heart attack*, or a short text such as *an obstruction of the blood supply to the heart*. The goal is to assign m with a concept c. Formally, given a list of preidentified concept mentions $M = \{m_1, m_2, \ldots, m_n\}$ in the text and an ontology with a set of concepts $C = \{c_1, c_2, \ldots, c_t\}$, the goal of CN is to find a mapping function $c_j = f(m_i)$ that maps each textual mention to its correct concept.

We approach CN in 2 steps (see Figure 1): we first use a candidate generator $G(m, c)$ to generate a list of candidate concepts C_m for each mention m where $C_m \subset C$, and $|C_m| \ll |C|$. We then use a candidate ranker $R(m, C_m) \rightarrow \widehat{C_m}$, where $\widehat{C_m}$ is a re-ranked list of candidate concepts sorted by their relevance. But unlike information retrieval tasks where the order of candidate concepts in the sorted list $\widehat{C_m}$ is crucial, in CN we care only that the one true concept is at the top of the list.

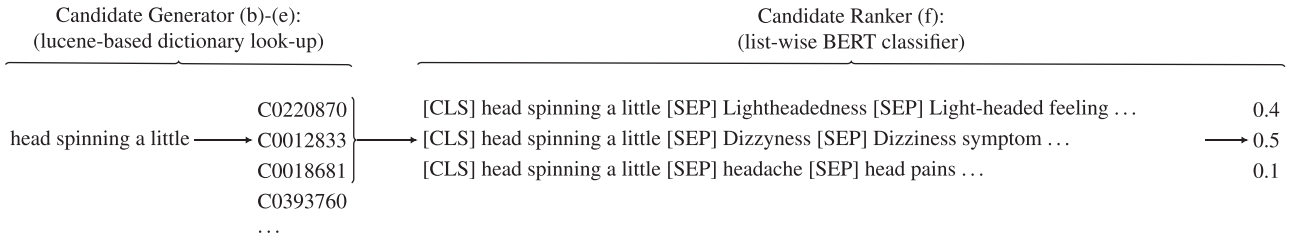

Figure 2. Outputs of Lucene(b-e) in *the* candidate generator are fed as inputs into the candidate ranker(f). BERT: Bidirectional Encoder Representations from Transformers.

The main idea of the 2-step approach is that we first use a simple and fast system with high recall to generate candidates, and then a more precise and discriminative system to rank the candidates.

Candidate generator

Multipass sieve systems[10,18] achieve competitive performance, though they only generate candidate concepts that are morphologically similar to the input mention. Inspired by the work in Luo et al,[10] we implement a Lucene-based sieve normalization system that consists of the following components:

1. Lucene index over the training data finds all concept unique identifier (CUI)-less mentions that exactly match mention m.
2. Lucene index over the training data finds CUIs of all training mentions that exactly match mention m.
3. Lucene index over UMLS finds CUIs whose preferred name exactly matches mention m.
4. Lucene index over UMLS finds CUIs where at least 1 synonym of the CUI exactly matches mention m.
5. Lucene index over UMLS finds CUIs where at least 1 synonym of the CUI has high character overlap with mention m. To check the character overlap, we run the following 3 rules sequentially: token-level matching, fuzzy string matching with a maximum edit distance of 2, and character 3-gram matching.

If Lucene(b-e) generates C_m such that $|C_m| > 1$, then mention m and C_m are fed as input to the candidate ranker(f).

Candidate ranker

After the candidate generator produces a list of concepts, we use a listwise classifier(f) based on the BERT neural network[37] to select the most likely candidate. BERT allows us to match morphologically dissimilar (but semantically similar) mentions and concepts, and the listwise classifier takes both mention and candidate concepts as input. Because the candidate generator can generate any concept from UMLS, this pipeline can handle concepts that never appeared in the training data.

We use BERT similar to a question answering configuration: given a concept mention m, the task is to choose the most likely candidate concept c_m from candidate concepts C_m. As shown in Figure 2, our classifier input includes the text of the mention m and all the unique synonyms of the candidate concept c_m, and takes the form "$[CLS]m[SEP]Syn_1(c_m)[SEP]\ldots[SEP]Syn_s(c_m)[SEP]$," where $Syn_1(c_m)$ is the i_{th} synonym of concept c_m. We also experimented with only using the preferred terms but it resulted in worse performance, likely because the synonyms provide useful additional lexical variants. (For instance, the preferred term for the concept mention "sinus tract" is "pathologic fistula," while its synonyms include "Fistula," "Abnormal sinus tract," "Sinus," "fistulous tract," etc.) For each such input, we extract BERT's final hidden vector $V_{(m,\ c_m)} \in \mathbb{R}^H$ corresponding to the first input token ("[*CLS*]"), and then concatenate all such vectors for all candidate concepts to form a matrix $V_{(m,\ C_m)} \in \mathbb{R}^{|C_m| \times H}$. We use this matrix and classification layer weights $W \in \mathbb{R}^H$, and compute a standard classification loss:

$$L_R = y * log(softmax(V_{(m,\ C_m)} W^T)) \quad (1)$$

where y is a one-hot vector, and $|y| = |C_m|$.

Semantic type regularizer. To encourage the listwise classifier toward a more informative ranking, we propose an ST that is optimized when candidate concepts with the correct UMLS semantic type are ranked above candidate concepts with the incorrect types. The semantic type of the candidate concept is assumed correct only if it exactly matches the semantic type of the gold truth concept. If the concept has multiple semantic types, all must match. Formally, we define:

$$R_p(\hat{y}_t, \hat{y}_p) = \sum_{p \in P(y)} m_1 + \hat{y}_p - \hat{y}_t \quad (2)$$

$$R_n(\hat{y}_p, \hat{y}_n) = \sum_{p \in P(y)} \max_{n \in N(y)} m_2 + \hat{y}_n - \hat{y}_p \quad (3)$$

where $\hat{y} = V_{(m,\ c_m)} W^T$, $N(y)$ is the set of indexes of candidate concepts with incorrect semantic types (negative candidates), $P(y)$ (positive candidates) is the complement of $N(y)$, $\hat{y}_t$ is the score of the gold truth candidate concept, and thus $\hat{y}_t \in P(y)$. The margins m_1 and m_2 are hyperparameters for controlling the minimal distances between $\hat{y}_t$ and $\hat{y}_p$ and between $\hat{y}_p$ and $\hat{y}_n$, respectively. Intuitively, R_p tries to push the score of the gold truth concept above all positive candidates at least by m_1, and R_n tries to push the best scored negative candidate below all positive candidates by m_2.

The final loss function we optimize for the BERT-based listwise classifier is:

$$L = L_R + \lambda R_p(\hat{y}_t, \hat{y}_p) + \mu R_n(\hat{y}_p, \hat{y}_n) \quad (4)$$

where λ and μ are hyperparameters to control the tradeoff between standard classification loss and the ST. Optimized with such a loss function, the BERT-based ranker is able to incorporate semantic type information into its predictions without requiring semantic types at prediction time.

Experiments

Evaluation metrics

In addition to the official evaluation metric (overall) accuracy, we also calculate the accuracy for *seen* concepts (C_{seen}) vs *unseen* concepts (C_{unseen}), that is, concepts that were vs were not seen during training. For the dev set, concepts in the training data are considered seen. For the test set, concepts in the training or dev data are considered seen. Formally,

$$Accuracy_{overall} = \frac{C_{predicted} \cap C_{gold}}{C_{gold}}$$

$$Accuracy_{seen} = \frac{C_{predicted} \cap C_{seen}}{C_{seen}}$$

$$Accuracy_{unseen} = \frac{C_{predicted} \cap C_{unseen}}{C_{unseen}}$$

where $C_{gold} = C_{seen} \cup C_{unseen}$.

Experimental design

We separate out the different contributions from the following components of our architecture.

1. Candidate generator: different combinations of components in Lucene(a-e), eg, indexing the training set, Lucene(a + b), or indexing components of UMLS, Lucene(c + d + e). When used alone, we take the first matched concept as the prediction.
2. BERT-based ranker(f): the listwise classifier, which always requires a candidate generator. During training, we experiment with Lucene(e) or Lucene(c + d + e) to generate training instances for the BERT-based classifier, with these candidate generators being run over the training set, and any mentions that have multiple matched candidate concepts becoming training instances. During prediction, we experiment with Lucene(e), Lucene(c + d +e) and Lucene(a + b + c + d + e) to generate candidates. For all experiments, we use BioBERT-base,[38] which further pre-trains BERT on PubMed abstracts (PubMed) and PubMed Central full-text articles. In our preliminary experiments, we also explored the Clinical-BERT,[39] but this resulted in worse performance than BioBERT. This finding is also shown in Ji et al.[44] We use huggingface's pytorch implementation of BERT (https://github.com/huggingface/transformers).
3. Semantic type regularizer: the ST, which always requires a BERT-based classifier f.

Submitted runs

For the shared task, when multiple candidate concepts were matched by Lucene components b, c, or d, they were sent to a rule-based reranker that considered section types, and concepts matched by Lucene component e were sent to the BERT-based listwise classifier. We also implemented an acronym matcher following component **e** that expanded the abbreviations and fed the expanded form as new input to component a. The BERT-based listwise classier f-cde was trained on examples from Lucene(c + d + e), and took the first 10 matched candidate concepts as input.

We submitted 3 system runs in this task. The first run used Lucene(a + b + c + d + e), the acronym matcher, and the rule-based reranker. The second and third runs used the same candidate generator as the first run, but paired it with f-cde. The second run used only 40 files to train f-cde, tuning hyperparameters on the 10 development files. The third run used all 50 files to train f-cde, but unfortunately was submitted with a row-alignment bug. The final submitted BERT-based listwise classier is an ensemble of 3 different training runs.

Postevaluation

After the shared task, we removed the acronym matcher and rule-based ranker, and used Lucene(e) instead of Lucene(c + d + e) to generate training instances for the BERT-based listwise classifier f-e as it generated many more training examples and resulted in better performance on the dev set. For the f-e, we take the first 30 matched candidate concepts as input, and we enhance f-e with an ST. Our final model f-e is a single training run. Figure 1 shows our complete architecture. We tune the hyperparameters via grid search, and select the best BERT hyperparameters based on the performance on dev set (see the Supplementary Material for the full hyperparameter details).

RESULTS

Table 2 shows the accuracy of multiple systems on the dev and test sets. The submitted run rows show that our Lucene-based lookup (submitted run #1) outperforms the previous state of the art,[10] 79.44% vs 76.35%, and our BERT-based ranker (submitted run #2) further improves performance to 81.66%. Comparing submitted run #1 and Lucene(a + b + c + d + e), which was the same but without the acronym matcher and rule-based ranker, we see that these 2 components make little difference (79.25% vs 79.44%), and hence we excluded them from the remaining experiments.

Looking at the Lucene-only rows, we can see that using only UMLS preferred terms—Lucene(c)—yields poor accuracy (34.35%) and low recall@30 (34.56%), while using all synonyms with either exact matching or partial matching—Lucene(d) or Lucene(e)—yields better accuracy (53.26% or 57.50%) and higher recall@30 (62.86% or 85.73%). However, we can also see that UMLS preferred terms have low coverage but high precision: when added to any pipeline, they improve that pipeline's accuracy while lowering its recall@30: Lucene(c + d) is more accurate than Lucene(d), and Lucene(c + d + e) is more accurate than Lucene(d + e), but the reverse is true for recall@30.

Comparing the Lucene + BERT rows to the Lucene-only rows, we see that adding the BERT-based ranker always improves performance. For example, while Lucene(e) achieves only 57.50% accuracy, adding BERT(f -e + ST) increases accuracy to 77.98%. Applying the same BERT model on top of Lucene(a + b + c + d + e) yields the best accuracy we achieved, 83.56%. Encouragingly, performance improves not only for seen concepts, but also for unseen concepts with all BERT models. This suggests that our generate-and-rank approach to CN is successful, and generalizes better than the Lucene-based lookup alone. In contrast to our results with Lucene alone, we find that specially handling preferred terms is no longer necessary with the BERT ranker: for example, the Lucene(e) + BERT(f -e + ST) that ignores preferred terms actually outperforms the Lucene(c + d + e) + BERT(f-e + ST) model that handles them (77.98% to 75.00%), probably in part because Lucene(e) has substantially higher recall than Lucene(c + d + e) (85.73% vs 79.21%).

Comparing the rows with and without the ST (+ST) we can see that encouraging the model to learn about UMLS semantic types leads to small but consistent gains across all experiments, with larger gains for unseen concepts than for seen ones. For example, on unseen concepts, Lucene(e) + BERT(f-e + ST) outperforms Lucene(e) + BERT(f-e): 64.01% to 62.26%.

DISCUSSION

Error analysis of the candidate generator

Table 3 shows the performance for each component in the candidate generator. Components run first have better performance, as is standard in a sieve-based system. We analyze the errors from each component:

Table 2. Accuracy of different systems on the dev and test sets

Systems		Dev				Test			
		Recall @30 (%)	Overall Accuracy (%)	Seen Accuracy (%)	Unseen Accuracy (%)	Recall @30 (%)	Overall Accuracy (%)	Seen Accuracy (%)	Unseen Accuracy (%)
Previous state of the art[10]			—	—	—		76.35	—	—
Submitted run	Submitted run #1	—	—	—	—		—	—	—
	Submitted run #2	—	—	—	—		—	—	—
	Submitted run #3	—	—	—	—		—	—	—
	Submitted run #3 (after fixing row-alignment bug)	—	—	—	—		—	—	—
Lucene only	Lucene(a + b)	52.30	52.07	77.00	0	55.08	54.66	77.91	0
	Lucene(c)	32.59	32.59	36.36	24.71	34.56	34.35	38.43	24.77
	Lucene(d)	62.22	53.48	58.49	43.02	62.86	53.26	57.20	43.98
	Lucene(e)	86.15	58.07	58.93	56.29	85.73	57.50	57.76	56.89
	Lucene(c + d)	58.74	55.56	60.35	45.54	60.06	56.43	60.95	45.82
	Lucene(d + e)	81.78	63.04	64.95	59.04	82.20	62.11	63.15	59.65
	Lucene(c + d + e)	78.30	65.11	66.81	61.56	79.21	65.29	66.90	61.49
	Lucene(a + b + e)	88.15.	76.67	86.64	55.84	88.38	77.43	86.23	56.75
	Lucene(a + b + c + d + e)	87.41	78.96	87.62	60.87	87.87	79.25	86.89	61.30
Lucene + BERT	Lucene(e) + BERT(f-e)	86.15	78.96	84.88	66.59	85.73	77.36	83.78	62.26
	Lucene(a + b + e)+ BERT(f-e)	88.15.	83.13	91.05	66.59	88.38	82.06	90.41	62.46
	Lucene(a + b + c + d + e) + BERT(f-e)	87.41	83.56	91.02	67.96	87.87	82.75	90.30	64.99
	Lucene(e) + BERT(f-e + ST)	86.15	79.85	85.10	68.88	85.73	77.98	83.92	64.01
	Lucene(c + d + e) + BERT(f-e + ST)	78.30	75.41	77.77	70.48	79.21	75.00	78.59	66.57
	Lucene(a + b + e)+ BERT(f-e + ST)	88.15.	84.05	91.71	68.05	88.38	82.90	90.90	64.10
	Lucene(a + b + c + d + e)+ BERT(f-e + ST)	87.41	84.44[a]	91.57	69.57	87.87	83.56[a]	90.80	66.52

Accuracy indicates how often the top prediction is correct (for the Lucene-only rows, we take the first matched concept as the top prediction). Recall@30 indicates how often the correct candidate is within the first 30 matched candidate concepts.

BERT: Bidirectional Encoder Representations from Transformers.

[a]The best performance.

Table 3. Accuracy for each component of the candidate generator in our best complete system Lucene(a + b + c + d + e) + BERT(f-e + ST) on dev set

	Overall			$\|C_m\| = 1$		$\|C_m\| > 1$		
Components	$\|m\|$	Accuracy (%)	Recall@30 (%)	$\|m\|$	Accuracy	$\|m\|$	Accuracy (%)	Recall@30 (%)
Lucene(a + b)	705	97.16	97.59	681	97.65	24	83.33	95.83
Lucene(c)	165	84.42	84.42	161	84.47	4	75	75
Lucene(d)	164	73.78	81.10	127	82.68	37	43.24	75.68
Lucene(e)	315	38.41	69.84	6	16.67	309	38.83	70.87
Lucene(a + b + c + d + e)	1350	78.96	87.41	976	92.93	374	42.51	72.99

Accuracy indicates how often the first matched candidate concept is correct. Recall@30 indicates how often the correct candidate is within the first 30 matched candidate concepts. $|C_m|$ indicates the size of the candidate concepts. |m| indicates number of mentions predicted by each component.

BERT: Bidirectional Encoder Representations from Transformers; ST: semantic type regularizer.

Table 4. Accuracies of our proposed architectures and their oracle versions on dev set

	Overall	$\|C_m\| > 1$		
System	Accuracy (%)	$\|m\|$	Accuracy	Recall@30 (%)
Lucene(a + b + c + d + e) + BERT(f-e + ST)	84.44	374	62.23	72.99
Lucene(a + b + c + d + e) + BERT(f-e + ST) (Oracle CandGen)	88.07	374	75.40	100
Lucene(e) + BERT(f-e + ST)	79.85	1327	80.11	86.51
Lucene(e) + BERT(f-e + ST) (Oracle CandGen)	89.19	1327	89.60	100

Accuracy indicates how often the first matched candidate concept is correct. Recall@30 indicates how often the correct candidate is within the first 30 matched candidate concepts. Oracle CandGen indicates that we artificially inject the correct concept into the candidate generator's list if it was not there when $|C_m| > 1$.

BERT: Bidirectional Encoder Representations from Transformers; ST: semantic type regularizer.

(**a + b**): Exact matching against the training data makes only 705 predictions, even though 913 concepts in the dev data were seen during training. The missing concepts are string mismatches between mentions and indexing, including different determiners such as "your incision" vs "the incision," abbreviations such as "an exercise tolerance test" vs "ETT," typos such as "dressing changes" vs "adressing change," prepositional phrases with different orders such as "debulking of tumor" vs "tumor debulking," etc. These missed matches demonstrate the fragility of simply memorizing the training data, and the importance of using other resources such as UMLS synonyms (which are able to predict more than 70% of these missing concepts). Most false positive errors come from ambiguous mentions where the same string is mapped to more than 1 concept, eg, "acute" could mean "sudden onset (attribute)" or "acute phase," "ANTERIOR MYOCARDIAL INFARCTION" could mean "Old anterior myocardial infarction" or "Acute Anterior Wall Myocardial Infarction." Extra context information would be required to disambiguate these mentions.

(**c**): Exact matching against UMLS preferred terms has a great potential for CN because of its 84.42% accuracy and fewer predictions with $|C_m| > 1$ than other components. Most false positive errors come from the lack of context information or domain knowledge. For instance, c finds a spatial concept "brachial" for mention "brachial" in "... 2+ brachial, 1+ radial ...," while the correct concept is "brachial pulse, function." There are still many gaps though between the terms the clinicians use and the preferred terms in UMLS.

(**d**): Using synonyms and lexical variants for each concept increases the search space, which makes it possible to find concepts whose preferred term is lexically dissimilar to the mention, for example, (d) finds concept "pansystolic murmur" for mention "holosystolic murmur" as one synonym of the concept exactly matches the mention. However, it also has the disadvantage of finding more than 1 candidate concept, for example, (d) finds 3 candidate concepts "Examination of reflexes," "Observation of reflex," and "Reflex action," in which all of them share the same synonym "reflexes." Additional information such as semantic type, other synonyms, or the context is required to select the most probable one.

(**e**): Among 315 mentions predicted by partially matching against the UMLS synonyms, accuracy is 38.41%, while recall@30 is 69.84%; 98.10% of mentions have more than 1 mapped candidate concept. Most false positive errors come from (1) small character overlaps between mention and concept synonyms, for example, mention "upgoing" with concept "upward," mention "acute cardiopulmonary process" with concept "acute pulmonary heart disease"; (2) abbreviations such as mention "C02" with "Carbon dioxide content measurement"; (3) lack of domain knowledge such as mention "an 80% stenosis" with concept "partial stenosis"; and (4) ambiguous mention such as "an ulcer" with concept "pressure ulcer" and "ulcer."

Candidate ranker

We can quantify the remaining error in the system by examining where there is still room for improvement. Because Lucene(a + b + c + d + e) achieves an recall@30 of 87.41 on the dev set, that is also the upper bound for the accuracy performance of the BERT-based ranker. Lucene(a + b + c + d + e) + BERT(f-e + ST) achieves accuracy of 84.44%, a 5.48% improvement over the Lucene accuracy of 78.96%, and 2.97% away from the upper bound. We can also ask what would happen if the candidate generator always included the correct concept somewhere in the top-k. Table 4 shows that even with a top-k list that's guaranteed to include the correct concept, Lucene(a + b + c + d + e) + BERT(f-e + ST) achieves only 88.07%, 11.03% away from the upper bound in this case of 100%. So we

CUI	Preferred Term	Semantic Type	CG	+f-e	+f-e + ST
C4303631	Amputation of right lower limb above knee	Therapeutic or Preventive Procedure	5	1	1
C0002691	Amputation above-knee (procedure)	Therapeutic or Preventive Procedure	2	6	2
C0585921	O/E - Amputated right above knee	Findings	1	2	3
C4510100	Amputation of bilateral lower limbs above knee	Therapeutic or Preventive Procedure	21	4	4
C0576404	Amputated above knee (finding)	Findings	3	8	5

Figure 3. Predictions for mention "a right above-knee amputation" and their rankings from the candidate generator (CG), candidate generator + BERT-based ranker + ST (+f-e +ST), and candidate generator + BERT-based ranker (+f-e). BERT: Bidirectional Encoder Representations from Transformers; CUI: concept unique identifier; CG: candidate generator; ST: semantic type regularizer.

CUI	Preferred Term	Semantic Type	CG	+f-e	+f-e + ST
C4302817	Structure of calf of right lower leg	Body Part, Organ, or Organ Component	6	7	1
C1446265	Right calf circumference	Finding	3	3	2
C2032437	Pain of right calf	Sign or Symptom	1	6	3
C0448482	Posterior crural muscle structure	Body Part, Organ, or Organ Component	19	15	4
C4510469	MRI of right calf	Findings	2	1	5

Figure 4. Predictions for mention "right calf" and their rankings. CG: candidate generator; CUI: concept unique identifier; MRI: magnetic resonance imaging; ST: semantic type regularizer.

can conclude that while there is a need to improve the candidate generator so that it can find the correct candidate for the 12.59% of mentions where it fails to do so, the BERT-based reranker also needs to learn more about ranking such mentions to close its 11.03% performance gap.

Semantic type regularizer

We conduct qualitative analysis to show the advantage of the ST. Figure 3 shows an example in which using ST (+f-e + ST) and not using ST (+f-e) makes a correct prediction. The candidate generator is not able to pick up the correct concept, as its synonyms have less character overlap with the mention "a right above-knee amputation" compared with other concepts such as "O/E—Amputated right above knee." However, the BERT-based ranker is able to compare "Amputation of right lower limb above knee" with the mention, and recognize that as a better match than the other options, reassigning it to rank 1.

Figure 4 shows an example that illustrates why the ST helps. The mention "right calf" needs to be mapped to the concept with the preferred name "Structure of calf of right lower leg," which has the semantic type of "Body Part, Organ, or Organ Component." The BERT-based classifier alone ranks 2 findings higher, placing the correct concept at rank 7. After the ST is added, the system recognizes that the mention should be mapped to an organ, and correctly ranks it above the findings. We also see the same pattern in Figure 3, in which using ST ranks concepts with semantic type "Therapeutic or Preventive Procedure" higher than the remaining concepts.

Limitations and future work

There are several ways that our system could be improved. First, our analysis shows that there is room for the candidate generator to improve, likely via better handling of abbreviations and other orthographic variations. Second, the BERT-based ranker is constrained by the small size of the training data, and could benefit from generating training instances directly from UMLS instead of just the annotated data. Third, our ST is limited to recognizing when 2 concepts share the exact same UMLS semantic type, and could be improved by incorporating other UMLS relations, like is-a and part-of, or by checking partial matching of the semantic types if these 2 concepts have multiple semantic types. Fourth, our BERT-based ranker only takes all the synonyms of the matched candidate concept as input but ignores the Lucene ranking and which of the synonyms was matched, information that may be beneficial to the ranker. Finally, our model ignores context completely, and though the mentions in the MCN corpus have low ambiguity, there a still a number of errors that could be solved by incorporating contextual information.

CONCLUSION

We propose a concept normalization framework consisting of a candidate generator and a listwise classifier based on BERT that incorporates UMLS preferred terms, synonyms, and semantic types. Our candidate generator, a Lucene-based sieve normalization system, combines dictionary lookup over training data, UMLS preferred terms and synonyms. Our BERT-based candidate ranker incorporates semantic types through a regularizer, and its formulation as a mention-concept classifier allows it to predict concepts never seen during training. Our best submission to the 2019 n2c2 shared task track 3, based on this generate-and-rank framework, ranked third of 33 systems.

FUNDING

This work was supported in part by National Institutes of Health grant R01LM012918 from the National Library of Medicine (Site PI: SB). Part of the computations were done in systems supported by the National Science Foundation under Grant No. 1228509. This work has been in part co-authored by UT-Battelle, LLC, under Contract No. DE-AC05-00OR22725 with the U.S. Department of Energy. The content is solely the responsibility of the authors and does not necessarily represent the official views of the National Institutes of Health, National Science Foundation, UT-Battelle, or the Department of Energy.

AUTHOR CONTRIBUTIONS

DX and SB designed all the experiments and drafted the manuscript. DX ran experiments and conducted analysis of the outputs. MG and JZ analyzed the dataset and conducted preliminary experiments. KB and EB provided suggestions and computational resources, and supervised part of the work. SB supervised all steps of the work. All authors read and approved the final version of the article.

SUPPLEMENTARY MATERIAL

Supplementary material is available at *Journal of the American Medical Informatics Association* online.

CONFLICT OF INTEREST STATEMENT

None declared.

REFERENCES

1. Fleuren WWM, Alkema W. Application of text mining in the biomedical domain. *Methods* 2015; 74: 97–106.
2. Gonzalez GH, Tahsin T, Goodale BC, *et al*. Recent advances and emerging applications in text and data mining for biomedical discovery. *Brief Bioinform* 2016; 17 (1): 33–42.
3. Liu F, Jagannatha A, Yu H. Towards drug safety surveillance and pharmacovigilance: current progress in detecting medication and adverse drug events from electronic health records. *Drug Saf* 2019; 42 (1): 95–7.
4. Rumshisky A, Ghassemi M, Naumann T, *et al*. Predicting early psychiatric readmission with natural language processing of narrative discharge summaries. *Transl Psychiatry* 2016; 6 (10): e921.
5. Bodenreider O. The Unified Medical Language System (UMLS): integrating biomedical terminology. *Nucleic Acids Res* 2004; 32 (90001): 267D–70D.
6. Kelly L, Goeuriot L, Suominen H, *et al*. Overview of the ShARe/CLEF eHealth evaluation lab 2013. *Lect Notes Comput Sci* 2014; 8685: 172–91.
7. Pradhan S, Elhadad N, Chapman WW, *et al*. SemEval-2014 Task 7: analysis of clinical text. In: proceedings of the 8th International Workshop on Semantic Evaluation 2014: 54–62.
8. Weissenbacher D, Sarker A, Magge A, *et al*. Overview of the Fourth Social Media Mining for Health (SMM4H) shared tasks at ACL 2019. In: proceedings of the Fourth Social Media Mining for Health (#SMM4H) Workshop and Shared Task; 2019: 21–30.
9. Sarker A, Gonzalez-Hernandez G. Overview of the Second Social Media Mining for Health (SMM4H) Shared Tasks at AMIA 2017. In: *CEUR Workshop Proceedings 2017*; 1996: 43–8.
10. Luo YF, Sun W, Rumshisky A. MCN: a comprehensive corpus for medical concept normalization. *J Biomed Inform* 2019; 92: 103132.
11. Lee HC, Hsu YY, Kao HY. AuDis: an automatic CRF-enhanced disease normalization in biomedical text. *Database (Oxford)* 2016; 2016: baw091.
12. Martins B, Couto FM. ULisboa: Recognition and Normalization of Medical Concepts. In: proceedings of the 9th International Workshop on Semantic Evaluation (SemEval 2015); 2015: 406–11.
13. Kate RJ. Normalizing clinical terms using learned edit distance patterns. *J Am Med Inform Assoc* 2016; 23 (2): 380–6.
14. Aronson AR. Effective mapping of biomedical text to the UMLS Metathesaurus: the MetaMap Program. *Proc AMIA Symp* 2001; 2001: 17–21.
15. Chute CG, Savova GK, Zheng J, *et al*. Mayo clinical Text Analysis and Knowledge Extraction System (cTAKES): architecture, component evaluation and applications. *J Am Med Inform Assoc* 2010; 17 (5): 507–13.
16. Zheng K, Vydiswaran VGV, Liu Y, *et al*. Ease of adoption of clinical natural language processing software: an evaluation of five systems. *J Biomed Inform* 2015; 58: S189–96.
17. Chapman WW, Nadkarni PM, Hirschman L, *et al*. Overcoming barriers to NLP for clinical text: the role of shared tasks and the need for additional creative solutions. *J Am Med Inform Assoc* 2011; 18 (5): 540–3.
18. D'Souza J, Ng V. Sieve-based entity linking for the biomedical domain. In: proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 2: Short Papers); 2015: 297–302.
19. Jonnagaddala J, Jue TR, Chang NW, *et al*. Improving the dictionary lookup approach for disease normalization using enhanced dictionary and query expansion. *Database (Oxford)* 2016; 2016: baw112.
20. Stevenson M, Guo Y, Al Amri A, *et al*. Disambiguation of biomedical abbreviations. In: proceedings of the Workshop in Current Trends in Biomedical Natural Language Processing; 2009: 71–9. doi: 10.3115/1572364.1572374
21. Savova GK, Coden AR, Sominsky IL, *et al*. Word sense disambiguation across two domains: Biomedical literature and clinical notes. *J Biomed Inform* 2008; 41 (6): 1088–100. doi : 10.1016/j.jbi.2008.02.003
22. Limsopatham N, Collier N. Normalising medical concepts in social media texts by learning semantic representation. In: proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers); 2016: 1014–23.
23. Jimeno Yepes A. Word embeddings and recurrent neural networks based on Long-Short Term Memory nodes in supervised biomedical word sense disambiguation. *J Biomed Inform* 2017; 73: 137–47.
24. Festag S, Spreckelsen C. Word sense disambiguation of medical terms via recurrent convolutional neural networks. *Stud Health Technol Inform* 2017; 236: 8–15.
25. Lee K, Hasan SA, Farri O, *et al*. Medical concept normalization for online user-generated texts. In: proceedings of the 2017 IEEE International Conference on Healthcare Informatics, ICHI 2017; 2017: 462–9.
26. Tutubalina E, Miftahutdinov Z, Nikolenko S, *et al*. Medical concept normalization in social media posts with recurrent neural networks. *J Biomed Inform* 2018; 84: 93–102.
27. Niu J, Yang Y, Zhang S, *et al*. Multi-task character-level attentional networks for medical concept normalization. *Neural Process Lett* 2019; 49 (3): 1239–18.
28. Luo Y-F. A hybrid normalization method for medical concepts in clinical narrative using semantic matching. *AMIA Jt Summits on Transl Sci Proc* 2019; 2019: 732–40.
29. Liu H, Xu Y. A deep learning way for disease name representation and normalization. *Lect Notes Comput Sci* 2018; 10619: 151–7.
30. Leaman R, Islamaj Dogan R, Lu Z. DNorm: Disease name normalization with pairwise learning to rank. *Bioinformatics* 2013; 29 (22): 2909–17.
31. Li H, Chen Q, Tang B, *et al*. CNN-based ranking for biomedical entity normalization. *BMC Bioinformatics* 2017; 18 (S11): 79–86.
32. Nguyễn TN, Nguyễn TM, Đặng TH. Disease named entity normalization using pairwise learning to rank and deep learning. 2018. https://eprints.uet.vnu.edu.vn/eprints/id/eprint/3200/1/TechnicalReport_Nguyen-ThanhNgan%20%281%29.pdf Accessed October 1, 2019.
33. Murty S, Verga P, Vilnis L, *et al*. Hierarchical losses and new resources for fine-grained entity typing and linking. In: proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers); 2018: 97–109.
34. Cao Z, Qin T, Liu TY, *et al*. Learning to rank: from pairwise approach to listwise approach. *ACM Int Conf Proc Ser* 2007; 227: 129–36.
35. Xia F, Liu T-Y, Wang J, *et al*. Listwise approach to learning to rank. In: proceedings of the 25th International Conference on Machine Learning; 2008: 1192–9.
36. Peters M, Neumann M, Iyyer M, *et al*. Deep contextualized word representations. In: proceedings of the 16th Annual Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies; 2018: 2227–37.
37. Devlin J, Chang M-W, Lee K, *et al*. BERT: pre-training of deep bidirectional transformers for language understanding. In: proceedings of the 17th Annual Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies; 2019: 4171–86.

38. Lee J, Yoon W, Kim S, *et al*. BioBERT: a pre-trained biomedical language representation model for biomedical text mining. *Bioinformatics* 2020; 36: 1234–40.
39. Alsentzer E, Murphy J, Boag W, *et al*. Publicly available clinical BERT embeddings. In: proceedings of the 2nd Clinical Natural Language Processing Workshop; 2019: 72–8.
40. Huang K, Altosaar J, Ranganath R. ClinicalBERT: modeling clinical notes and predicting hospital readmission. *arXiv*:1904.05342. 2019.
41. Si Y, Wang J, Xu H, *et al*. Enhancing clinical concept extraction with contextual embeddings. *J Am Med Informatics Assoc* 2019; 26 (11): 1297–304.
42. Peng Y, Yan S, Lu Z. Transfer learning in biomedical natural language processing: an evaluation of BERT and ELMo on ten benchmarking datasets. In: proceedings of the International Workshop on BioNLP Open Shared Tasks (BioNLP-OST); 2019: 58–65.
43. Li F, Jin Y, Liu W, *et al*. Fine-tuning bidirectional encoder representations from transformers (BERT)–based models on large-scale electronic health record notes: an empirical study. *JMIR Med Inform* 2019; 7: e14830.
44. Ji Z, Wei Q, Xu H. BERT-based ranking for biomedical entity normalization. arXiv:1908.03548. 2019.
45. Miftahutdinov Z, Tutubalina E. Deep neural models for medical concept normalization in user-generated texts. In: proceedings of the 57th Annual Meeting of the Association for Computational Linguistics: Student Research Workshop; 2019: 393–9.
46. Wei Q, Ji Z, Si Y, *et al*. Relation extraction from clinical narratives using pre-trained language models. *AMIA Annu Symp Proc* 2019; 2019: 1236–45.
47. Lin C, Miller T, Dligach D, *et al*. A BERT-based universal model for both within- and cross-sentence clinical temporal relation extraction. In: proceedings of the 2nd Clinical Natural Language Processing Workshop; 2019; 2: 65–71.
48. Uzuner Ö, South BR, Shen S, *et al*. 2010 i2b2/VA challenge on concepts, assertions, and relations in clinical text. *J Am Med Informatics Assoc* 2011; 18 (5): 552–6.