Fine-grained Concept Linking using Neural Networks in Healthcare

Jian Dai
School of Computing,
National University of Singapore
Singapore
daij@comp.nus.edu.sg

Meihui Zhang*
School of Computing,
Beijing Institute of Technology
Beijing, China
meihui_zhang@bit.edu.cn

Gang Chen
College of Computer Science,
Zhejiang University
Hangzhou, China
cg@zju.edu.cn

Ju Fan
School of Information/DEKE lab,
Renmin University of China
Beijing, China
fanj@ruc.edu.cn

Kee Yuan Ngiam
Department of Surgery,
National University Hospital
Singapore
kee_yuan_ngiam@nuhs.edu.sg

Beng Chin Ooi
School of Computing,
National University of Singapore
Singapore
ooibc@comp.nus.edu.sg

ABSTRACT

To unlock the wealth of the healthcare data, we often need to link the real-world text snippets to the referred medical concepts described by the canonical descriptions. However, existing healthcare concept linking methods, such as dictionary-based and simple machine learning methods, are not effective due to the word discrepancy between the text snippet and the canonical concept description, and the overlapping concept meaning among the fine-grained concepts. To address these challenges, we propose a Neural Concept Linking (NCL) approach for accurate concept linking using systematically integrated neural networks. We call the novel neural network architecture as the COMposite AttentIonal encode-Decode neural network (COM-AID). COM-AID performs an encode-decode process that encodes a concept into a vector, and decodes the vector into a text snippet with the help of two devised contexts. On the one hand, it injects the textual context into the neural network through the attention mechanism, so that the word discrepancy can be overcome from the semantic perspective. On the other hand, it incorporates the structural context into the neural network through the attention mechanism, so that minor concept meaning differences can be enlarged and effectively differentiated. Empirical studies on two real-world datasets confirm that the NCL produces accurate concept linking results and significantly outperforms state-of-the-art techniques.

CCS CONCEPTS

• **Information systems** → **Data cleaning**;

KEYWORDS

Healthcare, Fine-grained concept linking, Neural networks

ACM Reference Format:
Jian Dai, Meihui Zhang, Gang Chen, Ju Fan, Kee Yuan Ngiam, and Beng Chin Ooi. 2018. Fine-grained Concept Linking using Neural Networks in Healthcare. In *SIGMOD'18: 2018 International Conference on Management of Data, June 10–15, 2018, Houston, TX, USA.* ACM, New York, NY, USA, 16 pages. https://doi.org/10.1145/3183713.3196907

1 INTRODUCTION

Recent years have witnessed the rapid growth in volume of the loosely-coupled healthcare data, including the electronic health records (EHR), medical images, lab tests, and others. Cross-cutting healthecare data analytics such as disease progress analysis [19] and readmission prediction [18, 23] call for the data integration in terms of the medical concepts. Typically, these concepts are referred to by text snippets. As such, linking the snippets to their concepts becomes a critical step towards unlocking the wealth of the healthcare data.

Text snippets in a hospital database are typically entered by different clinicians over many years. Hence, various writing styles or standards are used; synonyms, acronyms, abbreviations, and simplifications are prevalent. As a result, text snippets may deviate significantly from the canonical descriptions of the concepts that they refer to. Consider some concrete real-world examples in Figure 1(a). The text snippet (i.e., query) "*ckd 5*" is used in place of "*chronic kidney failure, stage 5*", referring to the ICD-10-CM [1] concept (i.e., code) `N18.5`. Furthermore, "*symptomatic anemia from menorrhagia*" represents the concept `D50.0` whose canonical description is "*iron deficiency anemia secondary to blood loss*". The severe word discrepancy between a healthcare text snippet and the corresponding canonical concept description makes it difficult to link the text snippet to the concept according to the surface strings.

Moreover, the concepts organized by an ontology often have the semantic meaning overlapping, rendering the linking task even harder. Consider the ICD-10-CM concepts in the

*Meihui Zhang is the corresponding author.

SIGMOD'18, June 10–15, 2018, Houston, TX, USA

ACM ISBN 978-1-4503-4703-7/18/06.
https://doi.org/10.1145/3183713.3196907

[1] The International Statistical Classification of Diseases and Related Health Problems, 10th revision, published by the World Health Organization (WHO).

ontology fragment shown in Figure 1(b), the concepts D50.0, D53.0, and D53.2 are all related to the disease "*anemia*" according to their canonical descriptions listed in the table. As a matter of fact, the semantic meanings of some concepts are similar, making it difficult to distinguish the concepts from the semantic perspective.

Given its importance, concept linking in healthcare has attracted a great deal of interest. We broadly classify the existing healthcare concept linking techniques into three categories. The first category consists of the *dictionary-base* approaches, including MedLEE [16], MetaMap [1], cTAKES [37], and NOBLECoder [42]. They rely on the constructed dictionaries to accomplish the term-to-concept matching. The second category is constituted by the *machine learning based* approaches [41, 43] that use the expert annotated data to train a classifier/model, categorizing text snippets into different concepts. The third category is made up of the *combined* approaches [24, 27] that aggregate the linking results from multiple methods and reconcile them to produce the final linkage.

We use the examples shown in Figure 1 to illustrate that the aforementioned two challenges have not been well-addressed by the existing proposals in the first and second categories. [2] For the five queries shown in Figure 1(a), as a *dictionary-base* approach, NOBLECoder [42] is unable to link q_1, because the core word "*ckd*" is not included in the word-to-term dictionary; q_2 is linked to R52, owing to the word "*pain*"; q_3 is wrongly linked to N15.0, owing to the word "*nephropathy*", q_4 is linked to D50.9 because of the term "*iron deficiency anemia*"; q_5 is linked to N64.9 and N92.0 simultaneously because of the words "*anemia*" and "*menorrhagia*". Unfortunately, the actual concepts are not included in the results produced by NOBLECoder [42]. It is difficult for a dictionary to rigidly cover all the word-term and term-concept correspondences in a large-scale healtchcare database. As a *machine learning based* approach, the logistic regression (LR) [43] uses hand-crafted features to build a multi-class classifier. When only tens of concepts are considered, it correctly links q_1 and q_2 to N18.5 and R10.9 respectively, thanks to its sharing number feature and prefix feature. However, it wrongly links the other three queries. The hand-crafted features can be effective for some cases when the involved concept number is small. However, they cannot adapt to a large number of concepts (or a large dataset) when there are many new factors to consider.

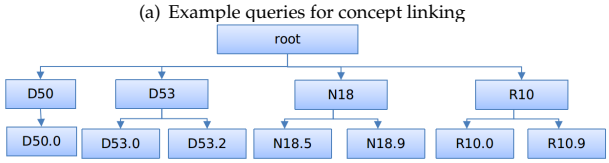
To address the limitations, we introduce a novel neural concept linking approach, called NCL. Instead of depending on a dictionary or some hand-crafted features, NCL leverages various concept descriptions extracted from a knowledge base (e.g., the UMLS [3]) and the concept hierarchy offered by an ontology (e.g., the ICD-10-CM included in the UMLS) to train the designed composite attentional encode-decode neural network (COM-AID) that synthesizes a numeric vector for each concept. The synthesized vector captures the semantic translations from a canonical description of a concept to various aliases of the concept, and embodies the hierarchal relationships between the concepts. Based on the synthesized vectors, COM-AID is capable of translating a concept into an arbitrary query, and evaluating the translation quality. In particular, COM-AID calculates the probability $p(q|c)$ of generating a text snippet q (e.g., a query) from a concept c, utilizing a customized *encode-decode process* (a.k.a. translation process). Moreover, unlike existing embedding techniques [25, 26, 31, 32] and attention mechanism [2] designed for text translation, COM-AID is equipped with a new attention mechanism designed for the concept-text translation.

In NCL, COM-AID is trained offline. In the online query processing, NCL first selects a few of concepts, leveraging a lightweight keyword matcher, and then ranks the concepts based on the translation probabilities computed by COM-AID. The top-1 concept is returned as the linked concept. Optionally, uncertain linked results can be forwarded to the domain experts for their feedbacks.

We implement NCL as part of Data Integration and Cleaning Engine (DICE) in GEMINI [28] [4] that provides a software stack for in-depth healthcare data analytics. Currently, GEMINI contains six subsystems: DICE ensures the data availability and quality, CDAS [12, 29, 33] exploits the crowd intelligence, epic [22] enables big data processing, SINGA [34] offers the distributed deep learning capability, CohAna [21] facilitates online data analysis, and iDat visualizes data. GEMINI has been used by NUHS [5].

ID	Query	Result
q_1	ckd 5	N18.5
q_2	abdomen pain	R10.9
q_3	ckd secondary to dm nephropathy	N18.9
q_4	chr iron deficiency anemia	D50.0
q_5	symptomatic anemia from menorrhagia	D50.0

(a) Example queries for concept linking



Concept	Text snippet
D50	Iron deficiency anemia
D50.0	Iron deficiency anemia secondary to blood loss
D53	Other nutritional anemias
D53.0	Protein deficiency anemia
D53.2	Scorbutic anemia

Concept	Text snippet
N18	Chronic kidney disease
N18.5	Chronic kidney disease, stage 5
N18.9	Chronic kidney disease, unspecified
R10	Abdominal and pelvic pain
R10.0	Acute abdomen
R10.9	Unspecified abdominal pain

(b) A disease ontology fragment

Figure 1: Some healthcare concept linking examples

To the best of our knowledge, this is the first neural-network based approach for concept linking in healthcare. In particular, we make the following contributions. 1) We vectorize the concepts organized by an ontology, leveraging its various descriptions extracted from a knowledge base and the concept hierarchy. 2) We softly match a vectorized concept to a

[2]The third category is not discussed, since the proposed approach can also be combined with other approaches in the *combined* approaches.
[3]https://www.nlm.nih.gov/research/umls/

[4]http://www.comp.nus.edu.sg/ dbsystem/gemini/index.html
[5]http://www.nuhs.edu.sg/

text snippet by an encode-decode process using the designed neural networks. 3) We design a novel composite attentional encode-decode neural network architecture (i.e., COM-AID), which incorporates two kinds of attentions into the decoder. Based on COM-AID, the probability of generating a text snippet from a concept can be computed. 4) We develop a neural concept linking framework (i.e., NCL), which leverages the trained COM-AID to conduct online concept linking. In addition, NCL is able to selectively collect feedbacks from domain experts. 5) We conduct empirical studies on two real-world datasets, which demonstrate that our proposal significantly outperforms NOBLECoder [42], the extended LR [43], pkduck [44], word move distance [25], Doc2Vec [26], sequence-to-sequence [40] and its variant [2] with attention mechanism.

The remainder of the paper is organized as follows. Section 2 covers basic concepts, problem statement, and reviews related work. Section 3 overviews our approach. Section 4 introduces COM-AID. Section 5 describes the online concept linking. Section 6 reports the experimental results. Section 7 concludes the paper.

2 PRELIMINARIES

This section presents preliminaries of our work. We formally define the problem of concept linking in Section 2.1, and review the related works in Section 2.2.

2.1 Problem Formulation

In general, *concept linking* matches unstructured text to the concepts in a knowledge base (KB). We formally define it as follows.

Concept. A concept $c = \{cid, d^c\}$, where *cid* is the unique identifier for c in KB, and d^c is a text snippets describing c. $d^c = \langle w_1^c, w_2^c, \cdots, w_n^c \rangle$ is a word sequence, where $w_i^c \in d^c$ is a word. We also call d^c the canonical description of c.

Concept Ontology. Our work considers a set of concepts in a KB, denoted by $\mathcal{C} = \{c_1, c_2, \cdots, c_m\}$, which are organized in a tree-structured *ontology* using the *sub-concept* relationships. More formally, an ontology $\mathcal{O} = \langle \mathcal{C}, \mathcal{E} \rangle$ is a tree structure with concepts in $\mathcal{C}$ as nodes and a set $\mathcal{E}$ of *sub-concept* relationships among the concepts (e.g., sub-classification of diseases). We use $\rightsquigarrow$ to denote a sub-concept relationship in $\mathcal{E}$. For instance, $c_1 \rightsquigarrow c_2$ represents that concept c_2 is a sub-concept of c_1.

Fine-grained concept. Given an *ontology* $\mathcal{O} = \langle \mathcal{C}, \mathcal{E} \rangle$, a concept c, when $c \in \mathcal{C}$ and $c \rightsquigarrow$ nil, is a fine-grained concept. That is, a concept without any sub-concepts is defined as a fine-grained concept.

Figure 1(b) shows an example ontology of disease concepts[6]. In this ontology, concepts are labeled with their *cid*s, such as `D50`, `D50.0`, and `R10.9`, which are also called ICD-10-CM codes. Moreover, concepts `D53.0` and `D53.2` are sub-concepts of concept `D53`. This figure also provides canonical descriptions of the concepts in the ontology. For example, "*Iron deficiency anemia*" describes concept `D50`. In this ontology fragment, the concepts `D50.0`, `D53.0`, `D53.2`, `N18.5`, `N18.9`, `R10.0`, and `R10.9` are fine-grained concepts.

Query. We focus on linking *textual queries* to the fine-grained concepts in the ontology. Formally, a *query* q is defined as a sequence of words $q = \langle w_1^q, w_2^q, \cdots, w_m^q \rangle$ that refers to a concept $c^\star \in \mathcal{C}$ in ontology $\mathcal{O}$. For example, Figure 1(a) shows five example queries, such as "*ckd 5*" and "*abdomen pain*".

Concept Linking. Based on the aforementioned notations, we define the problem of concept linking as follows.

Definition 2.1 (Concept Linking). Given an ontology $\mathcal{O} = \langle \mathcal{C}, \mathcal{E} \rangle$ and a query q, it aims to identify a fine-grained concept $c^\star \in \mathcal{C}'$ that most likely generates q. That is, $c^\star = \arg\max_{c \in \mathcal{C}'} p(q|c)$ where $p(q|c)$ is the probability of generating q from the concept c, $\mathcal{C}' \subseteq \mathcal{C}$ denotes the set of all fine-grained concepts.

Next, we provide an example of healthcare concept linking.

Example 2.2 (Disease Concept Linking). Figure 1 gives an example of disease concept linking, where Figures 1(b) and 1(a) respectively shows an ontology of disease concepts and five example queries. Concept linking aims to identify the concept referred to by a query. For instance, q_1 refers to `N18.5`. However, from these examples, we can observe that the *word discrepancy* between a query and the canonical description of the referred concept can be severe (e.g., "*ckd 5*" vs. "*chronic kidney disease, stage 5*"; "*symptomatic anemia from menorrhagia*" vs. "*iron deficiency anemia secondary to blood loss*"). Further, the semantic meaning difference between the fine-grained concepts can be minor. For example, `R10.0` is similar to `R10.9`; both `D53.2` and `D50.0` have the meaning of `anemia`.

2.2 Related Work

Concept Linking in Healthcare. In healthcare domain, various annotation approaches [1, 16, 24, 27, 37, 41–43] have been proposed. We categorize existing approaches into dictionary-based annotators [1, 16, 37, 42], simple machine learning based annotators [41, 43], and combined annotators [24, 27].

Dictionary-based annotators such as NOBLECoder [42], MetaMap [1], MedLEE [16], and cTAKES [37], leverage the dictionary extracted from a healthcare knowledge base (e.g., UMLS) or the dictionary manually created to recognize the concepts in the text. As the domain-specific knowledge base is used, these methods exhibit good performance in some bio-medical text genres. However, since the bio-related terminology is constantly evolving and the real-world text snippets are highly noisy, building a comprehensive lexical dictionary is unfeasible. As such, when the words not included in the dictionaries appear frequently, dictionary-based annotators tend to produce unreliable concept linking results.

Simple/shallow machine learning based annotators [41, 43] depend on some hand-crafted features to build models that are able to classify the text into different medical concepts. These models act as the soft string matchers that can deal with some terms not included in the dictionary, and hence achieve good results for specific bio-medical concept linking tasks. However, they can hardly adapt to different bio-medical text genres because of the hand-crafted features. Further, when

[6]The ontology is extracted from the well-accepted International Classification of Diseases, 10th Revision, Clinical Modification (*ICD-10-CM*).

the number of involved concepts grows, their classification capability may drop sharply.

Combined annotators [24, 27] combine multiple annotators that may complement each other to improve the overall annotation quality. As a concept linking method, our proposed NCL can also be combined with the other annotators.

Domain-independent Concept Linking. Linking a text snippet describing a named concept/entity to an appropriate concept/entity in a general-purpose knowledge base is challenging due to the name variations [35, 38] (i.e., an entity often exhibits multiple mention forms, including abbreviations, simplified forms, and alternative spellings) and the entity ambiguity [35, 38] (i.e., a text snippet may match multiple entities/concepts, and tend to be polysemous).

A normal concept linking process is comprised of two chief steps [9]: preliminary matching, and disagreement resolution. First, a text snippet is linked to multiple entities/concepts by the designed similarity measures [30], rules [14], classifiers [36], or distances [11]. Then, the clustering process [3] or the crowdsourcing techniques [7, 13, 15] are utilized to minimize/reconcile the entity linking disagreements. However, for healthcare text snippets, owing to the existence of severe word discrepancies, some out-of-the-vocabulary words appear in the queries. As a result, the correct concept cannot be effectively retrieved in accordance with the surface strings, resulting in the failure of preliminary matching. Moreover, in most cases, the clinicians are capable of correctly linking the text snippets to the concepts, rather than the crowd.

Recently, with the emergence of new techniques and data sources, researchers are able to match the text snippets in a more robust way, and reconcile the disagreement by leveraging some advanced inference techniques. To address the name variation challenge, the advanced approximate string matching algorithms [44] have been devised, which can also be applied to the healthcare concept linking. Meanwhile, some approaches [8, 39, 45] leverage external resources (such as WordNet, Wikipedia and term taxonomy) to expand the querying text snippets. These general-purpose external data sources are effective for the domain-independent text (e.g., Twitter posts). However, in this paper, we focus on linking the medical text snippets to the fine-grained concepts in the healthcare domain. To address the entity ambiguity challenge, some recent proposals [17, 35, 38, 46] link text snippets to the entities, assuming that a given text snippet can be directly found in a knowledge base, and leverage the surrounding entity contexts to jointly decide which entity should be linked to. Unfortunately, for the fine-grained healthcare concept linking problem, a text snippet only corresponds to a fine-grained concept, and no auxiliary entity context is available.

In fact, owing to the existence of severe word discrepancies and the concept meaning overlapping, the fine-grained concept linking is very challenging, and approximate string matching [44] seems a viable solution.

Text Comprehension. We consider healthcare concept linking from the text comprehension perspective, and propose to comprehend a query and the involved concepts before semantically linking them.

To comprehend a piece of text, each contained word can be modeled by a numeric vector, using the word embedding technique [31, 32]. In [25], based on the word embedding technique, Word Mover's Distance (WMD) is devised to measure the dissimilarity between a querying document and another document. More naturally, a piece of text can be comprehended by modeling the whole paragraph/document as a numeric vector (e.g., Doc2Vec [26]). Recently, due to its end-to-end framework, the sequence-to-sequence technique [40] is considered as the state of the art in machine translation. Further, its variant [2] improves the translation quality between two word sequences by introducing an attention mechanism.

Nevertheless, for healthcare concept linking, we need to link a text snippet to a concept. Due to the name variations [35, 38], a concept tends to be described by different text snippets. None of the existing machine translation methods consider how to model a concept with multiple descriptions and located in an ontology.

We note that GRAM [6] utilizes the concept co-occurrence relations to model the concepts, for the sake of addressing the data insufficiency issue. However, for healthcare concept linking, we aggregate the semantic meanings from the upper-level concepts into the corresponding most fine-grained concepts to disambiguate them.

3 NEURAL CONCEPT LINKING

We propose a *neural concept linking* (NCL) approach to facilitate concept linking. The basic idea of NCL is to utilize neural networks to perform an *encode-decode* process. The *encode-decode* process has achieved superior performance in many applications, such as machine translation [5] and reading comprehension [10]. As for concept linking, we observe that a concept can be described by various text snippets, as long as the core meaning of the concept is retained to a certain degree such that the text snippet will not be wrongly attributed to the other concepts. As such, we employ an *encoder* to encode each concept in the ontology into a *hidden state*, which is essentially a numerical vector learned for capturing the core meaning of the concept. We call the numerical vector *concept representation*. Then, given a query and a candidate concept, instead of measuring textual similarity between the query and the concept, we devise *decoders* to compute the probability that *decode*s the query from the hidden state of the concept. Based on this customized encode-decode process, for a query, NCL finds the concept with the maximum decoding probability as the concept linking result.

Figure 2 provides an overview of our neural concept linking (NCL) approach. At the heart of NCL, the neural network model COM-AID realizes the aforementioned encode-decode process. NCL consists of three components: 1) the offline MODEL TRAINING component trains the parameters of COM-AID using a large scale of training data; 2) the online CONCEPT LINKING component links queries to concepts in the ontology based on COM-AID; 3) the FEEDBACK CONTROLLER component collects feedback from domain experts (e.g., clinicians in the healthcare domain) for the uncertain

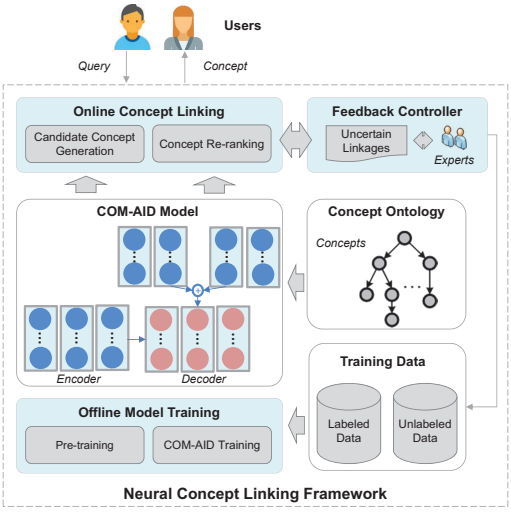


Figure 2: Overview of neural concept linking

linkages derived from the online concept linking. We shall describe the model and components below.

The COM-AID model. The model takes any concept $c \in \mathcal{C}$ in ontology $\mathcal{O}$ and a query q as input, and computes the probability $p(q|c)$ of generating q from the concept c. To this end, it makes use of an encoder and a decoder, as described below.

1) The *encoder* is essentially a function $h_{\text{ENC}} : c \in \mathcal{C} \rightarrow \mathbb{R}^d$ that maps a concept $c \in \mathcal{C}$ to a d-dimension numerical vector, denoted by $\mathbf{h}^c$. We materialize the function h_{ENC} in a recurrent neural network (RNN). The basic idea is to consider d^c as a word sequence $\langle w_1^c, w_2^c, \ldots, w_n^c \rangle$, and feed the word sequence to an RNN-based network so as to obtain a numerical vector $\mathbf{h}^c$ after sequentially encoding the whole word sequence.

2) The *decoder* takes hidden state $\mathbf{h}^c$ encoded for concept c and any query q as input, and measures the probability that q can be generated from $\mathbf{h}^c$. We consider that the larger the probability is, the more likely c is semantically related to q. Formally, the decoder is a function $f_{\text{DEC}} : \mathcal{Q} \times \mathbb{R}^d \rightarrow [0, 1]$ that maps a query $q \in \mathcal{Q}$ and a concept representation $\mathbf{h}^c$ into a probability within $[0, 1]$.

We employ an *attention*-based mechanism for decoding. First, given a query q, the decoder "attends" to the more relevant parts (i.e., sub-sequence) of d^c of concept c. For example, when q is "*abdomen pain*", decoder attends more on "*abdomen*" than "*unspecified*" for concept `R10.9` in Figure 1(b). Second, the decoder "attends" to the more relevant concepts, exploiting the ontology *structure*. Here, the ancestral concepts are considered. For instance, consider decoding query q_2 in Figure 1(a) from concept `R10.9`, the decoder also attends to its parent concept `R10`, which increases the probability for decoding word "*pain*" that is not contained in `R10.0`.

By systematically integrating the encoder and the decoder with attention mechanism into a neural network, NCL computes $p(q|c)$ of generating q from the concept c as

$$p(q|c; \Theta) = f_{\text{DEC}}(q, h_{\text{ENC}}(c)) \quad (1)$$

where Θ represents parameters in the encoder and the decoder. More details of COM-AID model are presented in Section 4.

Text snippet	cid
iron deficiency anemia secondary to blood loss	D50.0
anemia, chronic blood loss	D50.0
protein deficiency anemia	D53.0
amino acid deficiency anemia	D53.0
scorbutic anemia	D53.2

(a) Labeled text snippets

Unlabeled text snippet
scurvy
iron def anemia – from menorrhagia
fe def anemia 2' to menorrhagia
vitamin c def. anemia
orotaciduric anemia
symptomatic anemia from menorrhagia

(b) Unlabeled text snippets

Figure 3: Examples of training data for COM-AID

The Model Training component. NCL trains the COM-AID model using the *maximum likelihood estimation* approach. More specifically, the training component uses a set of *concept-query* pairs $\{\langle c, q \rangle\}$ (i.e., the labeled data) to train the model so that the learned concept representation corresponding to a concept can be decoded into various text snippets.

The *concept-query* pairs with respect to a concept c are instantiated by the canonical description of c and the alterantive descriptions (i.e., aliases) of c. As such, a training example regarding c can also be denoted by the pair $\langle d^c, d_j^c \rangle$, where d^c is the canonical description of c, and d_j^c is an alias of c. d_j^c comes from two sources: 1) it can be extracted from the knowledge base. For instance, in UMLS (the largest knowledge base in the healthcare domain), a concept may have different descriptions in different standards; take the concept `R10.0` as an example, it has the descriptions "*acute abdomen*", "*acute abdominal syndrome*", and "*pain; abdomen*"; 2) it can be a collected feedback regarding c, which will be described in the Feedback Controller component.

Further, NCL employs a *pre-training* component that learns word representations from massive *unlabeled text snippet*s (i.e., unlabeled data) and feeds the pre-trained results to COM-AID model. The unlabeled data come from two sources: 1) the queries, such as the accumulated notes written by the physicians in a hospital; 2) the labeled data with the incorporated concept information.

Figure 3 illustrates the two types of training data, and we shall elaborate how NCL utilizes the data for model training in Section 4.2.

The Concept Linking component. Given a query q, NCL is able to rigorously rank each fine-grained concept $c \in \mathcal{O}$ in descending order of probability $p(q|c; \Theta)$ and returns the one $c^\star$ with the largest probability as result. However, the computation of $p(q|c; \Theta)$ for all fine-grained concepts can be very expensive, especially for large knowledge bases. To address this issue, we introduce a two-phase online linking method. This method first retrieves a few top-ranking concepts according to a lightweight keyword matcher, and then re-ranks the retrieved concepts in accordance with their probability $p(q|c; \Theta)$ values computed by the trained COM-AID model. We present the details in Section 5.

The Feedback Controller component. Optionally, when NCL is unsure of the concept linking quality, the re-ranked concepts $\mathcal{C}'_k = \langle c_1, c_2, \cdots, c_k \rangle$ can be pooled for the domain experts' feedback. First, the uncertainty degree is evaluated for the re-ranked concepts. Then, based on the evaluated result, feedback controller determines if $\mathcal{C}'_k$ along with its query

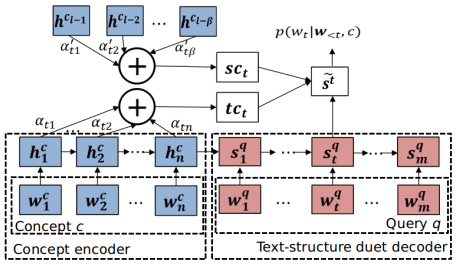


Figure 4: Network architecture of the COM-AID model

q should be pooled and presented to domain experts. Lastly, when sufficient feedbacks are obtained from the experts, feedback controller triggers a re-training process for COM-AID, and COM-AID will be re-trained by taking into account the newly collected feedbacks. As such, NCL learns from the domain experts and its concept linking ability is incrementally enhanced. Due to the space constraint, the details of feedback controller are presented in Appendix A.

4 THE COMPOSITE ATTENTIONAL ENCODE DECODE MODEL

This section presents our *composite attentional encode-decode* (COM-AID) model. We introduce network architecture of COM-AID in Section 4.1, and discuss how to train COM-AID in Section 4.2.

4.1 Neural Network Architecture

Figure 4 illustrates the network architecture of the COM-AID model. Intuitively, COM-AID takes a concept c and a query q as input, and produces a probability $p(q|c)$ of generating q from the concept. To this end, it consists of two neural networks. On one hand, the *concept encoder* (the left part of Figure 4) consumes the text snippet d^c of concept c, which is a sequence of words, denoted by $d^c = \langle w_1^c, w_2^c, \cdots, w_n^c \rangle$. The encoder utilizes an RNN-based network that encodes each word $w_t^c \in d^c$ into a hidden state $\mathbf{h}_t^c$. On the other hand, fed with the generated $\{\mathbf{h}_t^c\}$, the *text-structure duet decoder* (the right part of Figure 4) considers the query as a sequence of words, denoted by $q = \langle w_1^q, w_2^q, \cdots, w_m^q \rangle$, and measures how well $\{\mathbf{h}_t^c\}$ can be "decoded" into q by using a *text-structure attention mechanism* (as shown in the top part of Figure 4): 1) The decoder attends to the more relevant parts of $\{\mathbf{h}_t^c\}$ to capture the "core meaning" of d^c; 2) The decoder also attends to the concepts related to c by exploiting the ontology *structure*.

4.1.1 Concept Encoder. The concept encoder employs a Recurrent Neural Network (RNN) consisting of *Long Short-Term Memory (LSTM)* units to encode the word sequence d^c. In particular, each LSTM unit is used to encode one word w_t^c in d^c. The objective of a LSTM unit is to produce a hidden state $\mathbf{h}_t^c$ (i.e., a numeric vector) of the word w_t^c by considering both w_t^c and $\mathbf{h}_{t-1}^c$ generated from the previous unit, i.e.,

$$\mathbf{h}_t^c = h_{\mathsf{ENC}}(w_t^c, \mathbf{h}_{t-1}^c) \tag{2}$$

To be more specific, let $\mathbf{w}_t^c \in \mathbb{R}^{d\times 1}$ denote the embedding of word w_t^c. Let $\mathbf{W}^{(i)} \in \mathbb{R}^{d\times d}$, $\mathbf{W}^{(f)} \in \mathbb{R}^{d\times d}$, $\mathbf{W}^{(o)} \in \mathbb{R}^{d\times d}$, $\mathbf{W}^{(\tilde{e})} \in \mathbb{R}^{d\times d}$, and $\mathbf{U}^{(i)} \in \mathbb{R}^{d\times d}$, $\mathbf{U}^{(f)} \in \mathbb{R}^{d\times d}$, $\mathbf{U}^{(o)} \in \mathbb{R}^{d\times d}$, $\mathbf{U}^{(\tilde{e})} \in \mathbb{R}^{d\times d}$ be the weight matrices. Let $\mathbf{b}^{(i)} \in \mathbb{R}^{d\times 1}$, $\mathbf{b}^{(f)} \in \mathbb{R}^{d\times 1}$, $\mathbf{b}^{(o)} \in \mathbb{R}^{d\times 1}$, $\mathbf{b}^{(\tilde{e})} \in \mathbb{R}^{d\times 1}$ be bias vectors of the input gate $\mathbf{i}_t$, the forget gate $\mathbf{f}_t$, the output gate $\mathbf{o}_t$, and the state $\tilde{\mathbf{c}}_t \in \mathbb{R}^{d\times 1}$ for updating the memory cell in an LSTM unit, respectively. Let $\delta(\cdot)$ be the sigmoid function, $\tanh(\cdot)$ be the hyperbolic tangent function. Moreover, we use $\odot$ to denote the element-wise multiplication operation. The equations below computes the $\mathbf{h}_t^c$ which is the output hidden state of the t-th LSTM unit. For ease of presentation, we use $\mathbf{h}_t$ to denote $\mathbf{h}_t^c$ if the context is clear.

$$\begin{aligned}
\mathbf{i}_t &= \delta(\mathbf{W}^{(i)}\mathbf{w}_t^c + \mathbf{U}^{(i)}\mathbf{h}_{t-1}^c + \mathbf{b}^{(i)}) \\
\mathbf{f}_t &= \delta(\mathbf{W}^{(f)}\mathbf{w}_t^c + \mathbf{U}^{(f)}\mathbf{h}_{t-1}^c + \mathbf{b}^{(f)}) \\
\mathbf{o}_t &= \delta(\mathbf{W}^{(o)}\mathbf{w}_t^c + \mathbf{U}^{(o)}\mathbf{h}_{t-1}^c + \mathbf{b}^{(o)}) \\
\tilde{\mathbf{e}}_t &= \tanh(\mathbf{W}^{(\tilde{e})}\mathbf{w}_t^c + \mathbf{U}^{(\tilde{e})}\mathbf{h}_{t-1}^c + \mathbf{b}^{(\tilde{e})}) \\
\mathbf{h}_t &= \mathbf{o}_t \odot \tanh(\mathbf{e}_t)
\end{aligned}$$

Note that $\mathbf{w}_t^c$ is the *word representation* (a.k.a. word embedding) of $w_t^c \in d^c$, which can be initialized randomly or by our pre-train techniques introduced in Section 4.2.

By utilizing the LSTM unit described above, the text snippet $d^c = \langle w_1^c, w_2^c, \cdots, w_n^c \rangle$ is ultimately encoded into a numeric vector $\mathbf{h}^c = \mathbf{h}_n^c$ after the last word w_n^c is processed by the n-th LSTM unit, as shown in the box containing the symbol $\mathbf{h}_n^c$ in Figure 4. This $\mathbf{h}_n^c$ essentially captures all the information of text snippet d^c, and thus we consider $\mathbf{h}_n^c$ as the *concept representation* of concept c, which is then decoded in the decoder introduced below.

4.1.2 Text-structure Duet Decoder. The decoder takes the concept representation $\mathbf{h}^c$ and query q as input, and computes the probability $p(q|c)$ of generating q from concept c. Similar to the encoder, the decoder utilizes LSTM units to decode the word sequence $q = \langle w_1^q, w_2^q, \cdots, w_m^q \rangle$ of query q. Intuitively, it decomposes the computation of probability $p(q|c)$ as

$$p(q|c) = p(\langle w_1^q, \ldots, w_m^q \rangle | c) = \prod_{t=1}^{m} p(w_t^q | w_{<t}^q, c), \tag{3}$$

where $w_{<t}^q$ denotes the sequence of words before word w_t^q, i.e., $w_{<t}^q = \langle w_1, w_2, \ldots, w_{t-1} \rangle$.

Intuitively, the interpretation of $p(w_t^q | w_{<t}^q, c)$ is the probability of generating word w_t^q from its textual context $w_{<t}^q$ and concept c. In order to compute $p(w_t^q | w_{<t}^q, c)$, we introduce a vanilla LSTM-based method, which is presented as below.

A vanilla LSTM method for computing $p(w_t^q | w_{<t}^q, c)$. Let $\mathbf{s}_t^q$ denote the hidden state of the t-th LSTM unit in the decoder, which is illustrated as the red box in Figure 4. Similar to Equation 2, we can represent the computation of $\mathbf{s}_t^q$ as

$$\mathbf{s}_t^q = f_{\mathsf{DEC}}(w_t^q, \mathbf{s}_{t-1}^q), \tag{4}$$

where $f_{\mathsf{DEC}}(\cdot)$ is a standard LSTM unit introduced in Section 4.1.1. In particular, the initial state $\mathbf{s}_0^q$ is set as $\mathbf{h}_n^c$, which is the output of our encoder, as illustrated in Figure 4.

Based on Equation (4), a straightforward method for computing probability $p(w_t^q|w_{<t}^q, c)$ is to simply apply a `softmax` function over hidden state $\mathbf{s}_t^q$. However, this method may have the following limitations. First, it fails to differentiate the various parts of d^c when decoding query word w_t^q. For example, when decoding word "*abdomen*" query q_2 from concept `R10.9` (see Figure 1), the two words "*acute*" and "*abdomen*" should have different weights. Second, it does not consider the "structural context" of concept c. Intuitively, if q can be linked to concept c, its words should have larger chance to be decoded from the ancestor concepts of c (e.g., more general diseases). To address the limitations, we propose to incorporate the attention mechanism to the LSTM-based method.

LSTM with attention mechanism for computing $p(w_t^q|w_{<t}^q, c)$. The idea of our attention mechanism is to consider both *textual* and *structural* context of concept c. Following this idea, the prediction of w_t^q leverages a composite decoder state which contains the information derived from both the encoder states (i.e., the textual context), and the relevant concept representations (i.e., the structural context).

1) *Text-based Attention.* We consider that the decoding process of the t-th word w_t^q in the query should "attend" to *the most relevant part* of the text snippet d^c of concept c. To formalize this idea, when decoding query word w_t^q, we introduce α_{tr} as the weight of attending to the r-th hidden state $\mathbf{h}_r^c$ encoded from concept c. Following the attention mechanism introduced in [2], we compute this weight as

$$\alpha_{tr} = \frac{\exp(e_{tr})}{\sum_{p=1}^{n}\exp(e_{tp})} = \frac{\exp(\mathbf{h}_r^c \cdot \mathbf{s}_t^q)}{\sum_{p=1}^{n}\exp(\mathbf{h}_p^c \cdot \mathbf{s}_t^q)}, \tag{5}$$

where e_{tr} denotes the relatedness between $\mathbf{h}_r^c$ of concept c and $\mathbf{s}_t^q$ of query q, and it is computed by the inner product of $\mathbf{h}_r^c$ and $\mathbf{s}_t^q$.

Then, to aggregate the effect of all hidden states $\langle \mathbf{h}_1^c, \mathbf{h}_2^c, \cdots, \mathbf{h}_n^c \rangle$ on decoding word query w_t^q, we introduce a *textual context vector* $\mathbf{tc}_t$, and derive $\mathbf{tc}_t$ from hidden states $\{\mathbf{h}_t^c\}$ as

$$\mathbf{tc}_t = \sum_{r=1}^{n} \alpha_{tr}\mathbf{h}_r^c. \tag{6}$$

For example, considering the illustrative networks in Figure 4, when decoding the t-th word w_t^q in the query, COM-AID attends to hidden states $\{\mathbf{h}_1^c, \mathbf{h}_2^c, \ldots, \mathbf{h}_n^c, \}$ in the left part of the figure. For each hidden state $\mathbf{h}_r^c$, the model computes e_{tr} as the inner product of $\mathbf{h}_r^c$ and $\mathbf{s}_t^q$, and average all e_{tr}s to compute α_{tr}. Finally, the states $\{\mathbf{h}_1^c, \mathbf{h}_2^c, \ldots, \mathbf{h}_n^c, \}$ are summed up using α_{tr} as weights to generate $\mathbf{tc}_t$ for text-based attention.

2) *Structure-based Attention.* The structure-based attention considers the *structural context* of a fine-grained concept c in the ontology $\mathcal{O}$. The basic idea is to attend to concepts related to c and examine how well word w_t^q can be decoded from these concepts. In this paper, we consider the *ancestors* of c in the ontology.

Definition 4.1 (Structural Context). Given a depth β and a concept c_l, the structural context of c_l in the ontology $\mathcal{O}$ is a path $\mathcal{P} = \langle c_l, c_{l-1}, c_{l-2}, \cdots, c_{l-\beta}\rangle \subseteq \mathcal{O}$, where the concept $c_j \rightsquigarrow c_i$ if $j < i$. When $l < \beta$, the first level (except the root) concept is duplicated till the path length of $\mathcal{P}$ is equal to β.

Consider our example KB in Figure 1(b). Given a depth $\beta = 1$, the structural context of concept `D50.0` is $\langle$ `D50.0`, `D50` $\rangle$.

To formally incorporate the structural context defined above, we introduce a structural context vector $\mathbf{sc}_t$, which is computed from the hidden states $\langle \mathbf{h}^{c_{l-1}}, \cdots, \mathbf{h}^{c_{l-\beta}}\rangle$ corresponding to concept c's ancestors. Recall that hidden state $\mathbf{h}^{c_{l-r}}$ is encoded from concept c_{l-r} by our concept encoder introduced in Section 4.1.1. The structural context vector $\mathbf{sc}_t$ for the t-th word $w_t \in q$ is computed by a weighted sum of these concept representations.

$$\mathbf{sc}_t = \sum_{r=1}^{\beta} \alpha'_{tr}\mathbf{h}^{c_{l-r}} = \sum_{r=1}^{\beta} \mathbf{h}^{c_{l-r}} \frac{\exp(e'_{tl})}{\sum_{p=1}^{\beta}\exp(e'_{tp})} \tag{7}$$

where e'_{tr} scores how well the input concept representation $\mathbf{h}^{c_{l-r}}$ and state $\mathbf{s}_t^q$ of query word w_t^q matches. Similar to the previous textual context, e'_{tr} is computed by the inner product of $\mathbf{h}^{c_{l-r}}$ and $\mathbf{s}_t^q$.

To combine textual and structural contexts, COM-AID constructs a composite vector $[\mathbf{s}_t^q; \mathbf{tc}_t; \mathbf{sc}_t]$ by concatenating $\mathbf{s}_t^q$, $\mathbf{tc}_t$, and $\mathbf{sc}_t$. Then, it introduces an additional layer that takes as input the composite vector, and computes a vector $\widetilde{\mathbf{s}}_t$ as follows.

$$\widetilde{\mathbf{s}}_t = \tanh(\mathbf{W}_d[\mathbf{s}_t^q; \mathbf{tc}_t; \mathbf{sc}_t] + \mathbf{b}_d). \tag{8}$$

where $[\mathbf{s}_t^q; \mathbf{tc}_t; \mathbf{sc}_t] \in \mathbb{R}^{3d\times 1}$, $\mathbf{W} \in \mathbb{R}^{d\times 3d}$ is the weight matrix, and $\mathbf{b}_d \in \mathbb{R}^{d\times 1}$ is the bias vector for this layer.

Finally, leveraging the `softmax` function, the conditional probability $p(w_t^q|\mathbf{w}_{<t}^q, c)$ is computed as follows.

$$p(w_t^q|\mathbf{w}_{<t}^q, c) = \text{softmax}(\mathbf{W}_s\widetilde{\mathbf{s}}_t + \mathbf{b}_s) \tag{9}$$

where $\mathbf{W}_s \in \mathbb{R}^{|V|\times d}$ is the weight matrix, and $\mathbf{b}_s \in \mathbb{R}^{|V|\times 1}$ is the bias vector. $|V|$ denotes the vocabulary size.

4.2 Model Training

NCL adopts a *pretrain-and-refine* framework to train the COM-AID model, which consists of the following two phases.

Pre-training Phase: Word representation learning. As shown in Figure 4, the embedding/representation $\mathbf{w}_i$ of any word w is taken as input by both our encoder and decoder. There exist many ways to initialize the embedding vectors, such as giving random values, or training them using the existing word embedding learning methods [4, 31]. However, the existing methods follow the distributional hypothesis that words surrounding by the similar words share the similar meanings. However, this hypothesis is not exactly right for concept linking. This is because the concept mentions are normally very short, and the word co-occurrence may be misleading for differentiating concepts. For example, "*protein deficiency anemia*", and "*dietary folate deficiency anemia*", "*iron deficiency anemia unspecified*" refer to the concepts `D53.0`, `D52.0`, and `D50.0`, respectively. Applying the continuous bag-of-words model (CBOW) [31], the words "*protein*", "*folate*", and "*iron*" tend to

have similar word embeddings if these words seldom appear in other text snippets. However, these three words have totally different semantics.

To avoid the side effect of the distributional hypothesis, we propose to alter each text snippets by incorporating the available concept information (e.g., concept identifiers) into it, so as to differentiate the co-occurrence under different concepts. Note that the original unlabeled text snippets are unchanged. Specifically, given a window size α and a word sequence $\mathcal{W}$, the sub word sequence $\mathcal{W}' = \langle w_{i-\alpha}, \cdots, w_{i-1}, w_{i+1}, \cdots, w_{i+\alpha} \rangle \subseteq \mathcal{W}$ is the word context of $w_i \in \mathcal{W}$. After the incorporation, the word context of $w_i \in \mathcal{W}$ is $\mathcal{W}' = \langle w_{i-\alpha}, cid, w_{i-\alpha+1}, \cdots, w_{i-1}, cid, w_{i+1}, \cdots, w_{i+\alpha-1}, cid, w_{i+\alpha} \rangle$. For example, after the incorporation, "*protein deficiency anemia*", "*dietary folate deficiency anemia*", and "*iron deficiency anemia unspecified*" become "D53.0 *protein* D53.0 *deficiency* D53.0 *anemia*", "D52.0 *dietary* D52.0 *folate* D52.0 *deficiency* D52.0 *anemia*", and "D50.0 *iron* D50.0 *deficiency* D50.0 *anemia* D50.0 *unspecified*", respectively. As such, their word contexts are no longer similar. The incorporated concept information steers the word embeddings of "*protein*", "*folate*", and "*iron*" away from each other. After that, the word representations are learned by applying CBOW [31] to the altered text snippets.

Refinement Phase: COM-AID training using MLE. we construct the training data $\mathcal{D}$ consisting of $\langle d^c, d_j^c \rangle$ pairs for all the concepts, where d^c is the canonical text snippet of concept c and d_j^c is one of the labeled text snippet associated with the concept c (cf. Figure 3(a)). Further, we define the objective function as the sum of the log-likelihood of generating the $\langle d^c, d_j^c \rangle$ pairs from the COM-AID model. More formally, the objective function of training is:

$$\mathcal{J}(\Theta) = -\frac{1}{|\mathcal{D}|} \sum_{\langle d^c, d_j^c \rangle \in \mathcal{D}} \log p(d_j^c | d^c; \Theta) \tag{10}$$

We adopt mini-batch Stochastic Gradient Descent (SGD) for updating the parameter values. For a training example $\langle d^c, d_j^c \rangle$, d^c is taken as input by COM-AID and d_j^c is taken as text snippet to be decoded. After computing the loss, text-structure decoder progressively back-propagates the error to the concept encoder, and their parameters are updated accordingly. Note that during the error back-propagation, the word embeddings and the concept representations in the neural networks are also updated.

By minimizing the objective function, we aim to find the parameters Θ along with proper word embeddings and concept representations that make the encode-decode (a.k.a. translation) between standard text snippet of concept c and its alternative descriptions most likely, with the help of textual context and structural context.

5 ONLINE CONCEPT LINKING

This section presents how NCL utilizes the well-trained COM-AID model to perform online concept linking. The basic idea is to introduce a conditional probability $p(c|q)$ to measure how each concept $c \in \mathcal{C}$ is related to query q, and then take the concept $c^\star$ with the maximum probability $p(c|q)$ as the result, i.e., $c^\star = \arg\max_c p(c|q)$. Formally, based on the Bayesian formula, we have

$$p(c|q) = \frac{p(q|c;\Theta)p(c;\Theta)}{p(q)} \propto p(q|c;\Theta)p(c;\Theta), \tag{11}$$

where $p(c;\Theta)$ is a prior probability.

For general cases, we consider this prior follows a uniform distribution (e.g., every concept c has the same $p(c;\Theta)$) and maximize the likelihood (i.e. MLE). Note that this can be extended to the cases that $p(c;\Theta)$ is not equal among different concepts. In such cases, the prior could be considered as an input and the maximum a posteriori probability (MAP) estimation could be used in place of MLE. For MLE cases, we have the following equation.

$$p(c|q) \propto p(q|c;\Theta). \tag{12}$$

Therefore, following Equation (12), a straightforward approach for concept linking first enumerates every concept in the ontology, computes $p(q|c;\Theta)$ based on the COM-AID model, and then picks the one with the largest $p(q|c;\Theta)$. However, this approach will incur large computation cost, because the computation of $p(q|c;\Theta)$ is a forward propagation process in COM-AID and many expensive matrix (vector) operations may get involved.

To improve the efficiency, we propose a two-phase approach for concept linking that first efficiently generates a small set of *candidate* concepts and then computes $p(q|c;\Theta)$ only for the candidates.

Phase I: Generating candidate concepts. We generate candidate concepts using keyword search. More specifically, we compute the cosine similarity between each concept c and query q with the TF-IDF weighting scheme, and then return the top-k concepts with the largest similarity as the candidates.

However, the keyword search method may produce many *false negatives* due to the use of abbreviations and the existence of typos. Consider an example query "*dm 1 with neuropaty*": the method cannot generate good candidate concepts, as "*dm*" is an abbreviation of "*diabetes mellitus*" and "*neuropaty*" is a typo of "*neuropathy*".

We address this issue by replacing each out-of-vocabulary word with its semantically nearest word in the vocabulary. We call this step by *query rewriting*. More specifically, let Ω denote the word vocabulary of concepts in the ontology. For each word $w \in q$, if $w \notin \Omega$, we substitute w with its *nearest* word in Ω, denoted by $w^\star$. Here, we rely on the word embeddings to measure the semantic closeness. Recall that COM-AID has a pre-training phase for word embeddings, which leverages both the labeled text snippets and the unlabeled ones. Therefore, the vocabulary Ω' for the word embeddings in COM-AID contains not only the words in Ω but also the words from the unlabeled text snippets. Thus, we can compute the nearest words of w in the embedding space, i.e.,

$$w^\star = \arg\max_{w'} \texttt{cosine}(\mathbf{w}', \mathbf{w}), \tag{13}$$

where cosine is the cosine similarity and $\mathbf{w}'$ ($\mathbf{w}$) is the embedding representation of word w' (w). Note that if w is not in Ω',

we will first look for its textually similar word in Ω' (e.g., using edit-distance), and then apply Equation (13). For instance, query "*dm 1 with neuropaty*" can be rewritten into "*diabetes 1 with neuropathy*", where "*dm*" is replaced by "*diabetes*"; and "*neuropaty*" is corrected.

Phase II: Finding top candidate using COM-AID. Given a set of candidate concepts generated from the previous phase, this phase evaluates probability $p(q|c;\Theta)$ for each candidate. In particular, the words appearing in both the canonical description and the query are temporarily removed, and then the corresponding $p(q|c;\Theta)$ is computed according to Equation 3. NCL returns the concept with the largest probability.

6 EXPERIMENTS

In this section, we report the concept linking experimental results over two real-world datasets based on two ontologies.

6.1 Experimental Setup

Ontologies: Two ontologies ICD-9-CM and ICD-10-CM are used, where the ICD-9-CM has 17,418 concepts (14,567 are fine-grained concepts), and the ICD-10-CM has 93,830 concepts (71,486 are fine-grained). UMLS knowledge base supports both ontologies.

Datasets: Two datasets hospital-x and MIMIC-III are used. hospital-x consists of 860,080 diagnosis descriptions along with their disease codes extracted from the database of National University Hospital (Singapore). For hospital-x, we use a diagnosis description as a query, and the corresponding ICD-10-CM disease code as its referred concept. MIMIC-III consists of 58,976 diagnosis descriptions extracted from MIMIC-III [7] which is a public database containing records of more than 40,000 ICU patients. For MIMIC-III, a diagnosis description is also treated as a query, and its ICD-9-CM code is treated as a concept.

Methods for comparison: Seven methods are considered: NOBLECoder (**NC**) [42] that is a recently proposed dictionary-based method in healthcare domain, the recently proposed approximate string join method **pkduck** [44], the Word Mover Distance (**WMD**) [25] established over the word embedding technique [31, 32], the paragraph vector distributed representation technique (a.k.a. **Doc2Vec**) [26], and the extended logistic regression (**LR**) [43] for healthcare text.

For the logistic regression method [43], we extend it to consider not only the textual features (i.e., character bigrams, prefix/suffix, sharing numbers, acronym) described in the original paper, but also the structural features devised by us. For a concept c, its structural features are obtained by applying the textual feature functions in [43] to the aggregated text snippet of its ancestors' canonical descriptions. We use $\mathbf{LR}^+$ to denote the extended method.

Additionally, the sequence-to-sequence technique [40], and the machine translation method [2] that conducts the alignment and translation simultaneously, are also considered. Since these two methods [2, 40] can be viewed as the special cases of COM-AID, we only compare them with COM-AID in the network architecture study.

[7] https://mimic.physionet.org/

Parameters	Values
k	10, **20**, 30, 40, 50
β	1, **2**, 3, 4
d	50, 100, **150**, 200

Table 1: Parameter settings

Quality metrics: Two metrics are considered. First, the *top-1 accuracy rate* measures how much of the referred concepts appear in the first places of re-ranked concept lists. It is also called *accuracy* for short. Second, the *mean reciprocal rank* (MRR) $MRR = \frac{1}{|Q|}\sum_{i=1}^{|Q|}\frac{1}{rank_i}$ considers the specific location of the referred concept in the re-ranked concept list, where $rank_i$ denotes the rank position of the referred concept for the i-th query in the whole query set Q.

Training data: The training data contain the labeled text snippets extracted from UMLS, and the unlabeled text snippets from the knowledge base and the real-world datasets[8]. For fairness, the manually annotated text snippets (i.e., feedbacks) are not used for training COM-AID in the experiments presented in this section. In total, there are 194,094 labeled text snippets (concept aliases except the canonical descriptions for ICD-10-CM concepts) for hospital-x, and 176,736 labeled text snippets (concept aliases except the canonical descriptions for ICD-9-CM concepts) for MIMIC-III. [9] In total, there are 1,148,004 unlabeled text snippets for hospital-x, and 253,130 unlabeled snippets for MIMIC-III.

Note that the labeled data extracted from knowledge base is used to train COM-AID, and the performance of NCL is tested on real-world datasets (i.e., hospital-x and MIMIC-III).

Queries: We extract text snippets from hospital-x and MIMIC-III and use them as the queries. For each dataset, 484 queries are packed into a group, and the average accuracy/MRR values computed from 10 groups are reported. 84 purposely selected queries are contained in every group to cover different cases (e.g., abbreviation, synonym, acronym, and simplification); the rest are randomly chosen. The queries are generated as described above, unless otherwise stated.

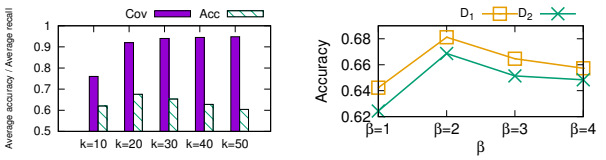
Parameters: We vary the dimensionality d of word/concept representation [10], the concept path length β, the online retrieved candidate concept set cardinality k, as shown in Table 1, where the default values are bolded.

Implementation: The neural networks are implemented in C++, compiled by GCC 5.4.1 under Ubuntu 12.04.4. The experiments are executed on a server with 503 GB memory and four E7540 CPUs.

[8] The labeled text snippets are also turned into the unlabeled data, when their concept information is incorporated into their content, as discussed in Section 4.2.

[9] Note that we have converted all the words into their lowercases,removed the special characters (e.g., ',', and ';'), and eliminated the duplicate text snippets. The corresponding canonical descriptions are excluded because one canonical description and one alias constitute a training example (e.g. ⟨acute abdomen, abdominal syndrome⟩), and a training example like ⟨acute abdomen, acute abdomen⟩ does not contribute to the COM-AID model.

[10] In COM-AID, the dimension of word representation can be different from the dimension of concept representation. For simplicity but without losing much generality, we assume that their dimensions are the same.



(a) Varying k (b) Varying β

Figure 5: Parameter study

6.2 Parameter Tuning

We tune the number of retrieved concepts k and the concept path length β, to evaluate the performance of NCL.

Varying k. We vary k from 10 to 50. A larger k leads to more fine-grained concepts retrieved in online concept linking phase I. We use 'Cov' to denote the average coverage value for hospital-x and MIMIC-III and 'Acc' to denote the average accuracy value. Figure 5(a) shows the computed results when k changes. Here, we define the queries whose retrieved concept lists in the online concept linking phase I that contain the referred concepts as Q'. Consequently, the coverage is defined as the ratio between $|Q'|$ and $|Q|$. As k increases, more concepts are retrieved. Hence, Cov grows. In the beginning, Acc grows, nevertheless, Acc slightly drops when $k > 20$. A larger k also incurs that more irrelevant concepts are retrieved in phase I. Thus, the concept linking quality in phase II is compromised. By default, $k = 20$ is used.

Varying β. We vary the concept path length β from 1 to 4 to study its effects on the concept linking accuracy. Figure 5(b) shows the experimental results. As β grows, initially, the computed accuracy over *hospital-x* and *MIMIC-III* increases; however, when $\beta > 2$, the accuracy begins to decline. This is because the ontology depths of ICD-9-CM and ICD-10-CM are typically less than 3 levels, and the duplication of top-level concepts does not help to improve the concept linking quality. We therefore use 2 as the default value of β.

6.3 Network Architecture Study

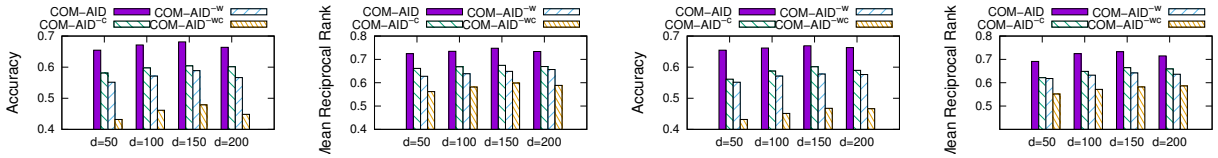
In this subsection, we evaluate the variants of COM-AID and discuss the necessity of systematically organizing two kinds of attentions in the text-structure duet decoder of COM-AID. In particular, three derived neural network architectures, namely COM-AID $^{-c}$, COM-AID $^{-w}$, and COM-AID $^{-wc}$, are studied. COM-AID $^{-c}$ removes the structural context based attention in COM-AID, and is an instance of the attentional neural network [2]. COM-AID $^{-w}$ discards the textual context based attention mechanism in COM-AID. COM-AID $^{-wc}$ discards both the textual context based attention and the structural based attention, which becomes a sequence-to-sequence network [40]. As a result, two existing methods [2, 40] are included in the comparison. The computed *accuracy* and *MRR* values are reported in Figure 6.

Figure 6(a) and Figure 6(c) show the evaluated accuracy values over hospital-x and MIMIC-III, respectively. We can see that COM-AID significantly outperforms the COM-AID $^{-c}$, COM-AID $^{-w}$, COM-AID $^{-wc}$ across two datasets with different hidden dimensions. In particular, COM-AID outperforms COM-AID $^{-c}$. After the removal of structural context based attention, the concept linking accuracy averagely drops 0.08, suggesting that the structural context $\mathcal{SC}$ in COM-AID is important. Taking into consideration the concept path helps differentiate the fine-grained concepts. To explain why COM-AID outperforms COM-AID $^{-c}$, we report a concrete instance. For the query "*chr iron deficiency anemia*", after processed by the model learned from COM-AID $^{-c}$, it is linked to the concept `E61.1` whose description is "*iron deficiency*". From the word perspective, "*iron deficiency*" fits the query "*chr iron deficiency anemia*". However, when taking into account the concept path of `E61.1` and the concept path of `D50.0`, it is easy to infer that `D50.0` is more appropriate. This is because the parent concept of `D50.0` (i.e., `D50`) is described by the text "*iron deficiency anemia*", which is closely related to the query. In contrast, the description of `E61` (i.e., the parent concept of `E61.1`) is referred to as "*deficiency of other nutrient elements*", which is not directly related to the query. Without incorporating $\mathcal{SC}$ into its decoder, COM-AID's concept linking capability degrades.

COM-AID outperforms COM-AID $^{-w}$. After the removal of textual context based attention, the concept linking accuracy averagely drops 0.1, indicating that incorporating textual context $\mathcal{TC}$ into COM-AID is necessary. And word-level attention helps decode the query from the concept representation. To appreciate why COM-AID outperforms COM-AID $^{-w}$, let's consider a concrete example. The query "*end-stage renal failure*" is linked to the concept `N18.9` "*chronic kidney disease, unspecified*" according to the model trained from COM-AID $^{-w}$. In contrast, the correct concept `N18.6` has the description "*end stage renal disease*". Although `N18.9` and `N18.6` have the same parent concept, `N18.6` is more appropriate, owing to the detailed information "*end stage*". COM-AID $^{-w}$ wrongly links the query to `N18.9`, suggesting that the word-level correlation between the query and the description of `N18.6` has not been fully captured. Without incorporating $\mathcal{TC}$ into its decoder, COM-AID's concept linking capability degrades.

The motivation for adding the attention mechanism in COM-AID is to bring more information to the decoding phase, leveraging the textual context based attention to handle the word discrepancies and the structural context based attention to differentiate fine-grained concepts. According to the experimental results with respect to COM-AID $^{-wc}$ and COM-AID shown in Figure 6, the removal of both the $\mathcal{SC}$ and $\mathcal{TC}$, on average, leads to more than 0.2 accuracy drop. The significant accuracy drop confirms the effectiveness of adding these attentions into the COM-AID.

The computed *MRR* values are shown in Figure 6(b) and Figure 6(d). We can see that COM-AID significantly outperforms the variants of COM-AID (i.e., COM-AID $^{-c}$, COM-AID $^{-w}$, and COM-AID $^{-wc}$) in terms of *MRR*. A higher MRR value means that on average the correct concepts are ranked higher than the wrong ones. COM-AID owns the relatively large MRR values, suggesting that COM-AID is able to place the correct concepts ahead of the other retrieved candidate concepts. By contrast, the variants of COM-AID cannot rank the correct concept properly.



(a) Accuracy, hospital-x (b) MRR, hospital-x (c) Accuracy, MIMIC-III (d) MRR, MIMIC-III

Figure 6: Network architecture study

6.4 Overall Linking Quality Study

In this subsection, we compare the overall concept linking qualities between NCL and its competitors: **pkduck** [44], **WMD** [25], **Doc2Vec** [26], **NC** [42], the extended **LR** [43], namely **LR**$^+$.

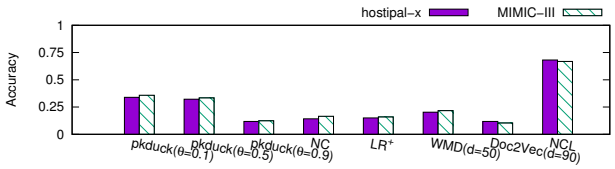
Accuracy comparison. We evaluate the methods' ability to correctly link the concepts over hospital-x and MIMIC-III, and report the *accuracy* values in Figure 7(a). Clearly, NCL produces the highest accuracy values over hospital-x and MIMIC-III, outperforming pkduck, NC, LR$^+$, WMD, and Doc2Vec by large margins. pkduck accounts for the second highest accuracy values, when its join similarity threshold $\theta = 0.1$. However, compared with the best accuracy of pkduck, NCL offers large accuracy enhancement.

The similarity threshold θ in pkduck is varied to gain insights into how the noisy unlabeled text snippets are joined with the labeled text snippets. From Figure 7(a), we can see that as θ decreases, the accuracy increases. This is because a smaller θ leads to more canonical concept descriptions approximately joined with the queries. However, a smaller θ also leads to a sharp growth of the joined entries, generating a lot of inappropriate text snippet pairs. Even when $\theta = 0.1$, the accuracy is below 0.34 for hospital-x and is less than 0.36 for MIMIC-III. Two reasons cause the low accuracy of pkduck. First, apart from the abbreviations, people tend to replace a word by its synonyms, and discard some inessential words. For instance, "*adenocarcinoma*" is employed in the query "*adenocarcinoma of colon*" instead of "*malignant neoplasm*". The pkduck similarity between the query and the concept description "*polyp of colon*" of the concept "*K63.5*" is 0.5; by contrast, the pkduck similarity between the query and the actually referred concept description "*malignant neoplasm of colon, unspecified*" of "*K18.9*" is 0.333. Second, the queries may contain many dangling words, incurring a higher pkduck value between two strings with many sharing words. For example, pkduck value between the query "*chr iron deficiency anemia*" and "*protein deficiency anemia*" is 0.4; however, the pkduck value between "*chr iron deficiency anemia*" and the actual referred concept description "*iron deficiency anemia secondary to blood loss (chronic)*" is 0.33. Therefore, the concepts whose canonical descriptions share many words with the query are considered to be better than the actually referred concept whose canonical description may only share a couple of essential words with the query. Typically, a query is short, making this issue more predominant.

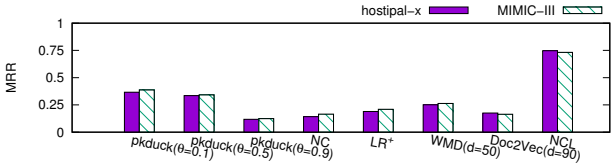
Two methods (i.e., NC and LR$^+$) from the healcare domain are compared with NCL. Figure 7 clearly show that both NC and LR$^+$ perform poorly on hospital-x and MIMIC-III. As a dictionary based method, NC relies on two hash tables (i.e., the word-to-term table and the term-to-concept table) to conduct concept linking according to the alignment of individual words. However, due to the severe word discrepancies between the queries and the canonical concept descriptions, NC tends to produce unreliable linking results. For example, the query "*exacerbation of eczema*" is linked to the concepts `L20.84` and `J47.1`, where the word "*eczema*" is contained in the canonical description of `L20.84`, and "*exacerbation*" is recognized by the description of `J47.1`. Instead, the actual concept should be `L30.9` whose canonical description is "*dermatitis, unspecified*". We count the linking result as correct if the actual concept appears in the concept list computed by NC. Even so, the accuracy of NC is very low, because the core words in a text snippet are often distorted and replaced by some other words that cannot be recognized by the word-to-term table. As a "simple" machine learning method, LR$^+$ depends on the hand-crafted features, including the textual features described in [43] and the structural features added by us, to build a multi-class classifier. Because the classifier amounts to a soft-string matcher, it is able to correctly link a query to its concept, when the querying text snippet is syntactically similar to its concept's description or the corresponding ancestral concept descriptions. However, as a multi-class classifier, the performance of LR$^+$ declines significantly as the number of considered concepts increases, suggesting that LR$^+$ is unable to scale with the number of concepts. Its accuracy values drop to nearly zero, when more than 30 fine-grained concepts are considered. Therefore, for LR$^+$, we limit the involved concepts to the candidate concepts retrieved by NCL. Even so, LR$^+$ performs poorly, suggesting that the features generated from the various surface string similarities are not suitable.

We vary the embedding dimension d for WMD to achieve the better accuracy over the datasets. When $d = 50$, the WMD achieves its relatively high accuracy over hospital-x and MIMIC-III. Nevertheless, the accuracy values are still small, suggesting that the word discrepancy compromises the effectiveness of word-level semantic distance. We also vary the embedding dimension d for Doc2Vec. When $d = 90$, Doc2Vec achieves its relatively better performance over hospital-x and MIMIC-III. Nevertheless, its accuracy values are less than 0.12, suggesting that the semantic overlapping between the fine-grained concepts makes the document-level semantic similarity difficult to distinguish them.

As for NCL, it has difficulty in comprehending the words that seldom appear. For example, it cannot distinguish the meaning difference between "*pancoast*" and "*testis*", leading to the wrong linking between the query "*left pancoast neoplasm*"



(a) Accuracy comparsion



(b) MRR comparsion

Figure 7: Performance comparsion

and the concept D29.22 whose canonical description is "*benign neoplasm of left testis*".

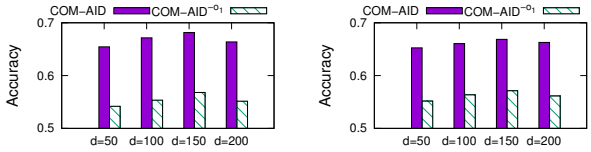
MRR Comparison. We further evaluate the effectiveness of different methods ranking the actually referred concepts. The computed MRR values are reported in Figure 7(b). Note that if the actually referred concept does not appear in the ranked/returned concept list, we ignore the corresponding $\frac{1}{rank_i}$ term. The absolute MRR values of NCL are the largest, indicating that NCL is able to place the correct concepts ahead of the other candidates.

When θ is large, pkduck is also capable of placing the correct concepts in the very front of the ranked concepts. We can see that when $\theta = 0.5$, the MRR values almost equal the corresponding accuracy values. This is because most of the top-1 concepts produced by pkduck are indeed the actually referred concepts. However, when θ is less than 0.5, the MRR values of pkduck are greater than its corresponding accuracy values, indicating that the correct concepts are not longer the top-1 concepts. The computed MRR values of NC, LR$^+$, WMD and Doc2Vec are all substantially lower than those of NCL across two datasets under different parameter settings, meaning that they are incapable of placing the correct concepts in the beginning of the returned concept lists.

6.5 Effect of Pre-training

We compare the concept linking accuracies derived from NCL and NCL without pre-training to gain insights into the design property of pre-training. The trained COM-AID model in the latter approach is denoted as COM-AID$^{-o_1}$.

Figure 8(a) and Figure 8(b) show the overall accuracy values derived from NCL using the models learned from COM-AID $^{-o_1}$ and COM-AID, respectively. We can see that as d grows, the computed accuracy also increases for both COM-AID and COM-AID $^{-o_1}$ when $d \leq 150$, and COM-AID produces significantly higher accuracy values than COM-AID $^{-o_1}$ for both hospital-x and MIMIC-III. The accuracy gap between COM-AID and COM-AID $^{-o_1}$ is consistently greater than 0.1, indicating that the proposed pre-training method is able to greatly enhance the concept linking quality under different settings. In particular, the incorporation of concept information into the text snippets leads to more accurate word representations



(a) Effect of pre-training, hospital-x (b) Effect of pre-training, MIMIC-III

Figure 8: Effect of pre-training

that helps differentiate the slight semantic meanings of two similar words, and results in more powerful COM-AID.

It may seem counter-intuitive that the accuracy values of hospital-x and MIMIC-III are similar, but the very short queries can weaken the encode-decode (i.e., translation) power of NCL, and hence reduce the accuracy. For instance, for MIMIC-III dataset, NCL has to translate the canonical descriptions "*hypertensive chronic kidney disease, malignant, with chronic kidney disease stage v or end stage renal disease*" to the queries "*hypertensive crisis*" and "*hypertensive urgency*", which is exetremely difficult; by contrast, for hospital-x dataset, NCL needs to link queries "*hypertension ef 75%*" and "*moderate pulmonary hypertension*" to concept I27.0 whose canonical description is "*primary pulmonary hypertension*", which is relatively easy. To some extent, the more fine-grained concepts in ICD-10-CM provide richer information, and hence help the translation between the canonical description of a concept and a query.

7 CONCLUSIONS

In this paper, we present a novel healthcare concept linking approach NCL that links the short and noisy real-world concept mentions to the fine-grained KB concepts in an ontology. The concept linking is accomplished by a translation (i.e., encode-decode) process supported by the devised COM-AID neural network. COM-AID leverages the attention mechanism to consider both the textual and structural contexts in the translation process. Experimental studies using two hospital datasets show that NCL produces accurate concept linking results and significantly outperforms seven state-of-the-art techniques.

We attribute NCL's good performance to three factors. First, owing to its powerful end-to-end semantic translation capability, the encode-decode process somehow eliminates the word discrepeancy between the surface strings. Second, a healthcare knowledge base contains various descriptions of a concept, offering the opportunity to capture the core meaning of the concept via self-translation. Third, the composite attention mechanism puts more information in the decoding phase, resulting in more accurate linking results.

ACKNOWLEDGMENTS

This research is funded by the National Research Foundation, Prime Ministers Office, Singapore, under its Competitive Research Programme (CRP Award No. NRF-CRP8-2011-08). Gang Chen was supported by the National Basic Research Program (973 Program No. 2015CB352400). Ju Fan was supported by the National Natural Science Foundation of China (NSFC) under Grant No. 61602488 and No. 61632016. We thank Jinyang Gao, Kaiping Zheng and the anonymous reviewers for their valuable comments and helpful suggestions.

REFERENCES

[1] Alan R Aronson and François-Michel Lang. 2010. An overview of MetaMap: historical perspective and recent advances. *Journal of the American Medical Informatics Association* 17, 3 (2010), 229–236.

[2] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473* (2014).

[3] Nikhil Bansal, Avrim Blum, and Shuchi Chawla. 2004. Correlation clustering. *Machine Learning* 56, 1-3 (2004), 89–113.

[4] Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Jauvin. 2003. A neural probabilistic language model. *Journal of machine learning research* 3, Feb (2003), 1137–1155.

[5] Denny Britz, Anna Goldie, Thang Luong, and Quoc Le. 2017. Massive Exploration of Neural Machine Translation Architectures. *ArXiv e-prints* (March 2017). arXiv:cs.CL/1703.03906

[6] Edward Choi, Mohammad Taha Bahadori, Le Song, Walter F Stewart, and Jimeng Sun. 2017. GRAM: Graph-based attention model for healthcare representation learning. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. 787–795.

[7] Gianluca Demartini, Djellel Eddine Difallah, and Philippe Cudré-Mauroux. 2012. ZenCrowd: leveraging probabilistic reasoning and crowdsourcing techniques for large-scale entity linking. In *Proceedings of the 21st international conference on World Wide Web*. ACM, 469–478.

[8] Bolin Ding, Haixun Wang, Ruoming Jin, Jiawei Han, and Zhongyuan Wang. [n. d.]. Optimizing index for taxonomy keyword search. In *Proceedings of the 2012 ACM SIGMOD International Conference on Management of Data*. 493–504.

[9] Xin Luna Dong and Divesh Srivastava. 2015. Big data integration. *Synthesis Lectures on Data Management* 7, 1 (2015), 1–198.

[10] Xinya Du, Junru Shao, and Claire Cardie. 2017. Learning to Ask: Neural Question Generation for Reading Comprehension. *arXiv preprint arXiv:1705.00106* (2017).

[11] Ahmed K Elmagarmid, Panagiotis G Ipeirotis, and Vassilios S Verykios. 2007. Duplicate record detection: A survey. *IEEE Transactions on knowledge and data engineering* 19, 1 (2007), 1–16.

[12] Ju Fan, Guoliang Li, Beng Chin Ooi, Kian-lee Tan, and Jianhua Feng. [n. d.]. icrowd: An adaptive crowdsourcing framework. In *Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data*. 1015–1030.

[13] Ju Fan, Meiyu Lu, Beng Chin Ooi, Wang-Chiew Tan, and Meihui Zhang. [n. d.]. A hybrid machine-crowdsourcing system for matching web tables. In *Data Engineering (ICDE), 2014 IEEE 30th International Conference on*. 976–987.

[14] Wenfei Fan, Xibei Jia, Jianzhong Li, and Shuai Ma. 2009. Reasoning about record matching rules. *Proceedings of the VLDB Endowment* 2, 1 (2009), 407–418.

[15] Tim Finin, Will Murnane, Anand Karandikar, Nicholas Keller, Justin Martineau, and Mark Dredze. 2010. Annotating named entities in Twitter data with crowdsourcing. In *Proceedings of the NAACL HLT 2010 Workshop on Creating Speech and Language Data with Amazon's Mechanical Turk*. Association for Computational Linguistics, 80–88.

[16] Carol Friedman, Hongfang Liu, Lyudmila Shagina, Stephen Johnson, and George Hripcsak. 2001. Evaluating the UMLS as a source of lexical knowledge for medical language processing.. In *Proceedings of the AMIA Symposium*. American Medical Informatics Association, 189.

[17] Octavian-Eugen Ganea, Marina Ganea, Aurelien Lucchi, Carsten Eickhoff, and Thomas Hofmann. 2016. Probabilistic bag-of-hyperlinks model for entity linking. In *Proceedings of the 25th International Conference on World Wide Web*. 927–938.

[18] Omar Hasan, David O Meltzer, Shimon A Shaykevich, Chaim M Bell, Peter J Kaboli, Andrew D Auerbach, Tosha B Wetterneck, Vineet M Arora, James Zhang, and Jeffrey L Schnipper. 2010. Hospital readmission in general medicine patients: a prediction model. *Journal of general internal medicine* 25, 3 (2010), 211–219.

[19] MJ Jeger. 2004. Analysis of disease progress as a basis for evaluating disease management practices. *Annu. Rev. Phytopathol.* 42 (2004), 61–82.

[20] Shihao Ji, SVN Vishwanathan, Nadathur Satish, Michael J Anderson, and Pradeep Dubey. 2015. Blackout: Speeding up recurrent neural network language models with very large vocabularies. *arXiv preprint* (2015).

[21] Dawei Jiang, Qingchao Cai, Gang Chen, HV Jagadish, Beng Chin Ooi, Kian-Lee Tan, and Anthony KH Tung. 2016. Cohort query processing. *Proceedings of the VLDB Endowment* 10, 1 (2016), 1–12.

[22] Dawei Jiang, Gang Chen, Beng Chin Ooi, Kian-Lee Tan, and Sai Wu. 2014. epiC: an extensible and scalable system for processing big data. *Proceedings of the VLDB Endowment* 7, 7 (2014), 541–552.

[23] Devan Kansagara, Honora Englander, Amanda Salanitro, David Kagen, Cecelia Theobald, Michele Freeman, and Sunil Kripalani. 2011. Risk prediction models for hospital readmission: a systematic review. *Jama* 306, 15 (2011), 1688–1698.

[24] Ioannis Korkontzelos, Dimitrios Piliouras, Andrew W Dowsey, and Sophia Ananiadou. 2015. Boosting drug named entity recognition using an aggregate classifier. *Artificial intelligence in medicine* 65, 2 (2015), 145–153.

[25] Matt Kusner, Yu Sun, Nicholas Kolkin, and Kilian Weinberger. 2015. From word embeddings to document distances. In *International Conference on Machine Learning*. 957–966.

[26] Quoc Le and Tomas Mikolov. 2014. Distributed representations of sentences and documents. In *Proceedings of the 31st International Conference on Machine Learning (ICML-14)*. 1188–1196.

[27] Ying-Chi Lin, Victor Christen, Anika Groß, Silvio Domingos Cardoso, Cédric Pruski, Marcos Da Silveira, and Erhard Rahm. 2017. Evaluating and improving annotation tools for medical forms. In *International Conference on Data Integration in the Life Sciences*. Springer, 1–16.

[28] Zheng Jye Ling, Quoc Trung Tran, Ju Fan, Gerald CH Koh, Thi Nguyen, Chuen Seng Tan, James WL Yip, and Meihui Zhang. 2014. GEMINI: an integrative healthcare analytics system. *Proceedings of the VLDB Endowment* 7, 13 (2014), 1766–1771.

[29] Xuan Liu, Meiyu Lu, Beng Chin Ooi, Yanyan Shen, Sai Wu, and Meihui Zhang. 2012. Cdas: a crowdsourcing data analytics system. *Proceedings of the VLDB Endowment* 5, 10 (2012), 1040–1051.

[30] Hans Peter Luhn. 1957. A statistical approach to mechanized encoding and searching of literary information. *IBM Journal of research and development* 1, 4 (1957), 309–317.

[31] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781* (2013).

[32] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*. 3111–3119.

[33] Beng Chin Ooi, Kian Lee Tan, Quoc Trung Tran, James WL Yip, Gang Chen, Zheng Jye Ling, Thi Nguyen, Anthony KH Tung, and Meihui Zhang. 2014. Contextual crowd intelligence. *ACM SIGKDD Explorations Newsletter* 16, 1 (2014), 39–46.

[34] Beng Chin Ooi, Kian-Lee Tan, Sheng Wang, Wei Wang, Qingchao Cai, Gang Chen, Jinyang Gao, Zhaojing Luo, Anthony KH Tung, Yuan Wang, et al. 2015. SINGA: A distributed deep learning platform. In *Proceedings of the 23rd ACM international conference on Multimedia*. 685–688.

[35] Delip Rao, Paul McNamee, and Mark Dredze. 2013. Entity linking: Finding extracted entities in a knowledge base. In *Multi-source, multilingual information extraction and summarization*. Springer, 93–115.

[36] Sunita Sarawagi and Anuradha Bhamidipaty. 2002. Interactive deduplication using active learning. In *Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*. 269–278.

[37] 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 (2010), 507–513.

[38] Wei Shen, Jianyong Wang, and Jiawei Han. 2015. Entity linking with a knowledge base: Issues, techniques, and solutions. *IEEE Transactions on Knowledge and Data Engineering* 27, 2 (2015), 443–460.

[39] Wei Shen, Jianyong Wang, Ping Luo, and Min Wang. 2012. Linden: linking named entities with knowledge base via semantic knowledge. In *Proceedings of the 21st international conference on World Wide Web*. ACM, 449–458.

[40] Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*. 3104–3112.

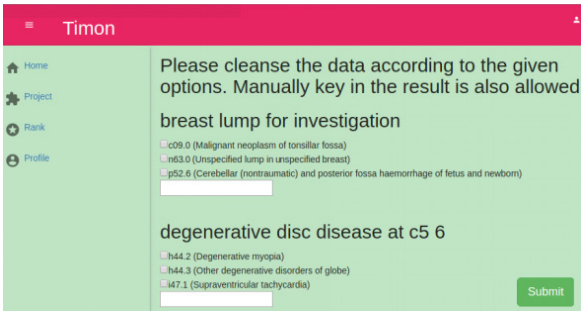
[41] Domonkos Tikk and Illés Solt. 2010. Improving textual medication extraction using combined conditional random fields and rule-based systems. *Journal of the American Medical Informatics Association* 17, 5 (2010), 540–544.

[42] Eugene Tseytlin, Kevin Mitchell, Elizabeth Legowski, Julia Corrigan, Girish Chavan, and Rebecca S Jacobson. 2016. NOBLE–Flexible concept recognition for large-scale biomedical natural language processing. *BMC bioinformatics* 17, 1 (2016), 32.

[43] Yoshimasa Tsuruoka, John McNaught, Jun'i; chi Tsujii, and Sophia Ananiadou. 2007. Learning string similarity measures for gene/protein name dictionary look-up using logistic regression. *Bioinformatics* 23, 20 (2007), 2768–2774.

[44] Dong Deng Wenbo Tao and Michael Stonebraker. 2018. Approximate String Joins with Abbreviations. In *Proceedings of the 44th international conference on Very large data bases*, Vol. 11. VLDB Endowment, 53–65.

[45] Wentao Wu, Hongsong Li, Haixun Wang, and Kenny Q Zhu. [n. d.]. Probase: A probabilistic taxonomy for text understanding. In *Proceedings of the 2012 ACM SIGMOD International Conference on Management of Data*. 481–492.



(a) Timon: A feedback collecting system

Query	breast lump for investigation	
Candidate	Canonical Description	Loss
C09.0	Malignant neoplasm of tonsillar fossa	18.1525
N63.0	Unspecified lump in unspecified breast	18.1605
P52.6	Cerebellar (nontraumatic) and posterior fossa haemorrhage of fetus and newborn	28.1594

(b) Input of Timon

Concept	Description
N63.0	unspecified lump in unspecified breast
N63.0	breast lump
N63.0	mammographic breast mass
N63.0	disorders of breast
N63.0	breast lump for investigation

(c) Output of Timon

Figure 9: Feedback controller workflow illustration

[46] Chenyan Xiong, Jimie Callan, and Tie-Yen Liu. 2017. Learning to attend and to rank with word-entity duets. SIGIR.

A FEEDBACK CONTROLLER IN NCL

Feedback controller controls the feedback collection, collect the feedbacks, and then re-train COM-AID model. In this fashion, NCL is able to incrementally enhance its concept linking capability.

A.1 Feedback Controller Component

We have developed a system (Timon) to realize the required functionalities. Figure 9 sketches the main workflow of Timon. Figure 9(b) shows an input example of Timon. It contains the query "*breast for investigation*", and the retrieved three candidate concepts `C09.0`, `N63.0`, and `P52.6`. The loss values in the third column are produced by COM-AID. We can see that first two loss values are close and all the losses are large, indicating that COM-AID is uncertain of the results. Therefore, NCL forwards the query along with the concepts and their canonical descriptions to Timon. When the specified number (e.g., 100) of the pooled uncertain queries has reached, Timon displays them in a generated web page, as shown in Figure 9(a). Note that the domain experts can either select a concept from the candidates or type a new concept in the blank text field. After collecting the feedbacks from the experts, Timon appends the newly collected feedbacks to the corresponding labeled training data. For example, Figure 9(c) indicates that the experts have selected the concept `N63.0` for the query "*breast lump for investigation*", because a new entry is appended to the descriptions of `N63.0`.

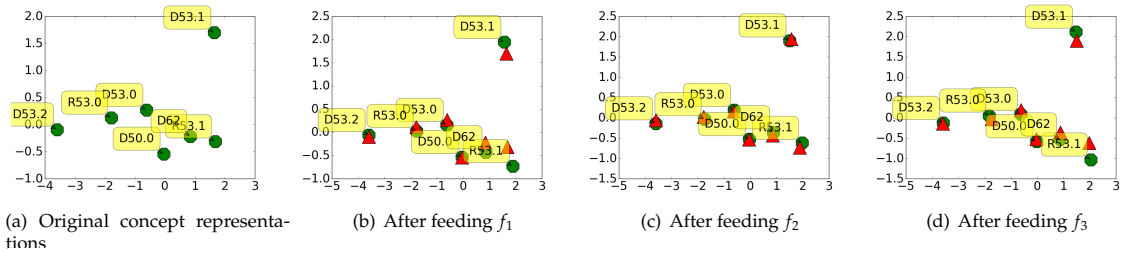
As illustrated, feedback controller examines the *uncertainty* of the re-ranked concepts and determines whether to solicit experts for feedbacks. Two factors are taken into account to evaluate the concept linking uncertainty based on the re-ranked concept list. First, the absolute probability value of the returned concept is considered. To avoid the floating-point underflow, The loss value $Loss = -\log p(q|c;\Theta)$ is used. When $Loss$ is high, NCL may produce inaccurate concept linkage. Second, the standard deviation of the loss value derived from the re-ranked list $\mathcal{C}_k$ is considered. We denote the standard deviation value by Std. A low Std suggests that the concepts in $\mathcal{C}_k$ own similar losses. Thereby, NCL is at the risk of producing inaccurate concept linking. When $Loss$ exceeds a predefined threshold or Std is lower than a predefined threshold, the feedback controller forwards q along with $\mathcal{C}_k$ to the concept linking pool, waiting for the domain expert's manual linking. Meanwhile, the current computed $c^\star = c_1$ is returned.

Moreover, the feedback controller is also responsible for examining the newly collected feedbacks. If the number of newly appended labeled training data entries exceeds a threshold, COM-AID will be re-trained by taking into account the newly collected feedbacks. This way, the concept linking capability of NCL is incrementally improved, and the domain experts can easily contribute their domain knowledge to NCL by their feedbacks.

A.2 Effect of Feedback Controller

As an end-to-end and data-driven model, COM-AID ensures that the collected feedbacks are effectively utilized. To investigate the effect of feedbacks, we perform incremental training. That is, after feeding one feedback (i.e., the labeled text snippet) into COM-AID, we train the model, take snapshots of the learned word representations and the concept representations, and assess the impact of the fed feedback.

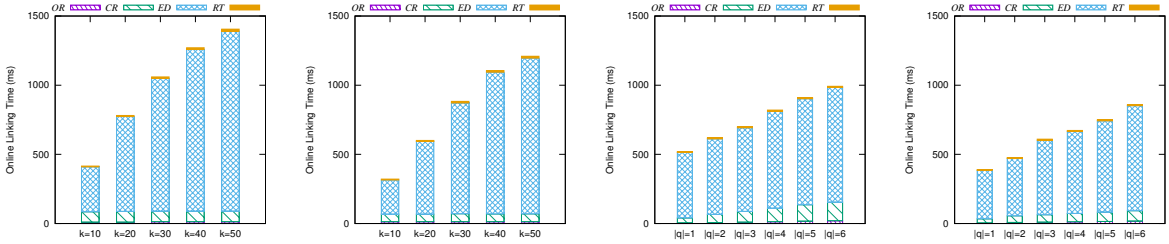
Figure 10(a)-10(d) show how the sampled word representations shift when incrementally feeding the COM-AID with expert feedbacks. In this experiment, feedbacks on three concept linking results (f_1, f_2, and f_3) are incrementally fed into COM-AID. f_1 = ⟨ `D50.0`, "*hemorrhagic anemia*" ⟩, f_2 = ⟨ `D62`, "*acute blood loss anemia*" ⟩, and f_3 = ⟨ `D53.2`, "*vitamin c deficiency anemia*" ⟩. Both green octagon and red triangle represent a concept representation projected by PCA. We use different markers for comparing the differences between two adjacent snapshots, where the red triangles represent the concept representations learned in the previous snapshot, and the green octagons represent the ones learned in current snapshot based on the current training data. We can see that after feeding f_1, the sampled concepts shown in Figure 10(b) all change their locations more or less. This is because adding one feedback into the training data also results in the change of word representations. While the concept representation of `D50.0` changes slightly, the ones corresponding to `D53.1` and `R53.1` significantly move away from `D50.0`, suggesting that adding f_1 to the training data makes the semantic meaning difference between `D50.0` and `D53.1` (or `R53.1`) larger. These semantic meaning differences are implied by the experts, because according to f_1, the word *hemorrhagic* from `D50.0` relates to "*blood*", and is irrelevant to "*weakness*" in `R53.1` and "*megaloblastic*" in `D53.1`. NCL successfully learns the semantic implication offered by the experts.



(a) Original concept representations (b) After feeding f_1 (c) After feeding f_2 (d) After feeding f_3

(e) Original word representations (f) After feeding f_1 (g) After feeding f_2 (h) After feeding f_3

Figure 10: Impact of Feedbacks



(a) hospital-x, varying k (b) MIMIC-III, varying k (c) hospital-x, varying $|q|$ (d) MIMIC-III, varying $|q|$

Figure 11: Online concept linking time analysis

Further, the learned semantic implication can be accumulated through more feedbacks. Comparing Figure 10(b) and Figure 10(c), we can see that the green octagons in current snapshot only shift slightly, compared with the locations of the red triangles. `D62` slightly moves away from `D50.0`, indicating the semantic difference between them becomes larger after feeding f_2. This move is caused by the word "*acute*" in f_2, which is different from the word "*chronic*" in the canonical description of f_1. As demonstrated, the learned semantic differences accumulate. The minor semantic implication in one feedback can be uncovered and learned by NCL, resulting in the slight shifts of the concept representations in the space. Comparing Figure 10(c) with Figure 10(d), adding f_3 into the training data leads to the shifts of `D53.1`, `D62`, `R53.1`, `R53.0`, and `R53.1`.

Figure 10(e)-10(h) show how the sampled word representations change when incrementally feeding the COM-AID with expert feedbacks. Comparing the word representations shown in Figure 10(e) and the ones in Figure 10(f), we can see that after feeding f_1 the distance between the word representation "*menorrhagia*" and "*blood*" becomes smaller, indicating that their semantic meanings get more similar. The word representation of "*anemia*" moves slightly away from the words "*acute*" and "*chronic*", because f_1 contains no information about "*anemia*" or "*acute*". Comparing Figure 10(f) and Figure 10(g), we can see that the word distance between "*anemia*" and "*acute*" gets smaller, because the added training example contains both "*acute*" and "*anemia*". After feeding f_3, as shown in Figure 10(h), the words "*anemia*" and "*vitamin*" move simultaneously toward the top right, meaning that the semantic meanings are affected by the added example "*vitamin c deficiency anemia*".

From these snapshots, we can see that the added training data drive the learned representations towards better semantic locations in the high-dimensional space. As such, the intricate semantic implications underlying the surface strings are grasped by NCL, from both the word and the concept perspectives.

B EFFICIENCY STUDY

We study both online and offline efficiency of NCL.

B.1 Online Linking Efficiency

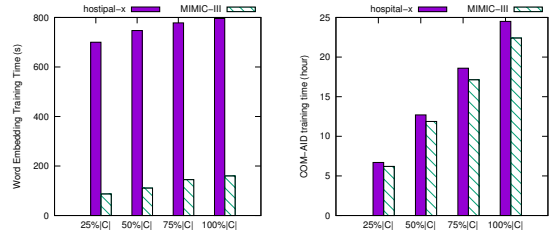
In this subsection, we report experimental online linking runtime when the candidate size k and the query length $|q|$ are varied. We vary k from 10 to 50, and vary $|q|$ from 1 to 6.

We divide the two online concept linking phases into four parts: the out-of-vocabulary word replacement (OR), the candidate concept retrieval (CR), the encode-decode process (ED), and the ranking (RT). For each query, we use one thread to perform OR, CR, and DT, and use ten threads to perform ED,

because after the candidates are obtained, their encode-decode processes can be executed separately.

Figure 11(a) and Figure 11(b) display the online concept linking times for different k values. As the size of candidate concept set increases, the running time of the online concept linking grows as well, which is mainly caused by the ED part. A larger k normally leads to more candidate concepts, invoking more encode-decode processes. Therefore, ED time increases. We also notice the sub-linear online linking time growth in Figure 11(a) and Figure 11(b), suggesting that the desired number of candidate concepts may not be met when k is large. The running time of hospital-x is larger than that of MIMIC-III, because we find that the canonical descriptions of the ICD-10-CM concepts are usually longer than those of the ICD-9-CM concepts. It takes more effort to encode a longer description and to compute the textual attention weights.

Figure 11(c) and Figure 11(d) show the running times for different query lengths. As the query length $|q|$ grows, the running time of the online concept linking grows as well. In particular, as $|q|$ grows, more postings in the inverted index are examined, resulting in the growth of CR time. Moreover, when the query is longer, more effort needs to be spent on the decoding and the attention computation. Thus, the ED time grows.



(a) Word embedding training time (b) COM-AID training time

Figure 12: Offline model training time analysis

B.2 Offline Training Efficiency

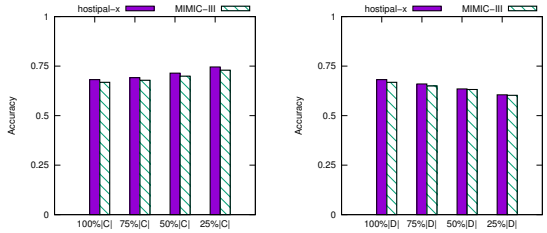
The offline training has a pre-training phase (i.e., the word embedding training) and a refinement phase (i.e., the COM-AID model training). We report the training time of the two phases in Figure 12(a) and Figure 12(b) respectively, owing to their different scales. We use 40 threads in the two phases.

As shown in Figure 12(a), the word embedding training times are short. It takes less than 800 seconds to train the word representations for hospital-x, and less than 170 seconds for MIMIC-III. Training data used here contain text snippets extracted from the knowledge base and all the text snippets of the corresponding real-world dataset. If a text snippet comes from the knowledge base, its concept information is incorprated into the text snippet. Otherwise, it remains unchanged. Clearly, the training time of hospital-x is larger than that of MIMIC-III. This is because hospital-x contains considerably more unlabeled training text snippets than MIMIC-III. Here, the parameter noise-contrastive estimation (NCE) is set to 10, the window size is 10, the iteration number is 10, and the learning rate is 0.05.

As shown in Figure 12(b), the COM-AID model training times occupy several hours. As the number of involved concepts increases, the training time in this phase grows as well. In this phase, however, the training time difference between two datasets is not very large. This is because COM-AID learns from the labeled text snippet pairs corresponding to the same concept (i.e., canonical description and alias pairs), and the amount of labeled training data for two datasets is similar. Further, in this phase, the training time growth is approximately linear, suggesting the good scalability of COM-AID training. The training time in this phase can be further reduced, when the BlackOut [20] technique is used.

C ROBUSTNESS EVALUATION

Experiments are conducted to study the effects of training data on the performance of NCL.



(a) Varying concept size (b) Varying unlabeled data

Figure 13: Varying training data

Varying Training Data. We perform two groups of experiments to investigate how the training data affects the performance of NCL. In the first group of experiments, for both the ICD-10-CM and ICD-9-CM, we vary the considered concept sizes from 25% to 100%. Accordingly, we extract the text snippets from the labeled data. For each concept set, we generate 500 queries whose actual concepts are covered to evaluate the performance. Figure 13(a) shows that the linking accuracy increases as the number of concepts drops. This occurs because a smaller concept size means less interfacing concepts, resulting in a higher accuracy value. Overall, the accuracy values change slightly over different concept sizes, suggesting that NCL is robust with respect to the labeled data.

In the second group of experiments, for both hospital-x and MIMIC-III datasets, we keep the labeled training data and the concepts unchanged, and vary the size of unlabeled data D from 25% to 100%. For each unlabeled data set, 500 queries are randomly generated from hospital-x and MIMIM-III, respectively. From Figure 13(b), we can see that the accuracy decreases as the unlabeled training data size drops. Nevertheless, when 25% unlabeled data are used, the computed accuracy values are still higher than 0.6, suggesting that NCL is able to maintain a high accuracy via the encode-decode process. Overall, the accuracy values change only slightly over different amounts of the unlabeled data, suggesting that NCL is robust with respect to the unlabeled data.