Mitigating Data Sparsity in Integrated Data through Text Conceptualization

Md Ataur Rahman, Sergi Nadal, Oscar Romero
Universitat Politècnica de Catalunya
Barcelona, Spain
md.ataur.rahman|sergi.nadal|oscar.romero@upc.edu

Dimitris Sacharidis
Université Libre de Bruxelles
Brussels, Belgium
dimitris.sacharidis@ulb.be

***Abstract*—We study the data sparsity problem for data generated from an integration system. We approach the problem from a textual information extraction perspective and propose to conceptualize external documents using the concepts in the integrated schema. We present THOR, a novel system that, unlike related approaches, neither relies on complex rules nor models trained with large annotated corpus, but on the integrated data and its schema without the need for human annotations. An extensive evaluation on the text conceptualization task demonstrates the superiority of our approach in terms of F1-score, effort and use of resources over the state-of-the-art language models.**

***Index Terms*—Data Integration, Information Extraction, Entity Recognition, Slot-filling**

I. Introduction

Data has always been an asset. Collecting, storing and integrating the available wealth of internal and external data enables significant improvements to the analytical capabilities of organizations. *Data integration* focuses on providing a unified view of data over a set of disparate and heterogeneous sources [1], and typically combines the underlying datasets with operators that allow for partial matches, such as *outer join* [2] or *full disjunction* [3]. The consequence, however, is the generation of a large number of missing values (a.k.a. *labeled nulls*, denoted by $\perp$), which are reported to account for 15% of the values [4], since each data source captures a different set of instances (particularly, if these datasets have been independently generated). Further, each data source provides a partial view of the data of interest, thus even after combining them the resulting integrated data presents an incomplete view. This is commonly referred to in the literature as the *data sparsity problem* [5] [6].

Missing value imputation is a very common data cleaning technique in the data integration pipeline to enhance data quality [7]. Focusing on repairing structured data, such cleaning approaches can be categorized into constraint-based and learning-based ones. Systems in the former category (e.g., LLUNATIC [8], NADEEF [9], or HORIZON [10]), aim to model and enforce data dependencies or business rules modeled as denial constraints [11]. Alternatively, those in the latter (e.g., HoloClean [12], Baran [13], or Garf [14]), automatically generate data repairs leveraging pre-trained probabilistic models or models trained from the available structured data. Despite their effectiveness, the approaches above are limited to

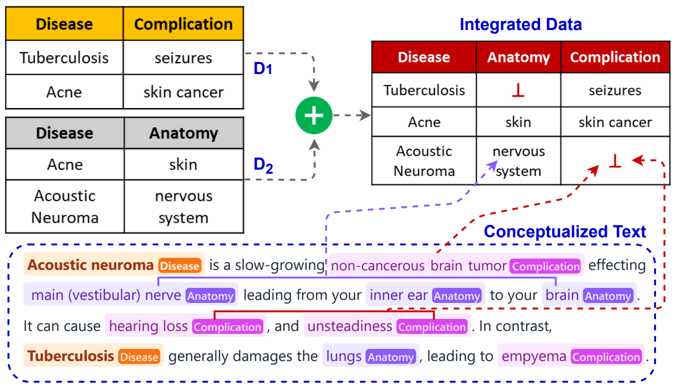

Disease
Complication
Tuberculosis
seizures
Acne
skin cancer
Disease
Anatomy
Acne
skin
Acoustic Neuroma
nervous system
D1
D2
Integrated Data
Disease
Anatomy
Complication
Tuberculosis
⊥
seizures
Acne
skin
skin cancer
Acoustic Neuroma
nervous system
⊥
Conceptualized Text
Acoustic neuroma Disease is a slow-growing non-cancerous brain tumor Complication effecting main (vestibular) nerve Anatomy leading from your inner ear Anatomy to your brain Anatomy.
It can cause hearing loss Complication, and unsteadiness Complication. In contrast,
Tuberculosis Disease generally damages the lungs Anatomy, leading to empyema Complication.

Fig. 1. An example illustrating the data sparsity problem and our solution through text conceptualization: missing values ($\perp$) regarding '*Complication*', and additional '*Anatomy*' information for '*Acoustic Neuroma*' are filled from the conceptualized text. Our approach leverages the knowledge embedded within unstructured text to enhance the completeness of integrated data.

the available structured sources to learn and compute repairs. However, as reported, a significant portion of data organizations hold resides in texts [15], which is largely unused. Hence, we approach the missing value imputation problem from a different perspective, and **aim to impute missing values over structured data leveraging on textual data**.

Fig. 1, exemplifies the scenario we address in the context of health-related data integration. Here, both datasets to be integrated (i.e., D_1 and D_2) contain the concept '*Disease*', but the instances available in each data source are different, thus, when combined, they generate a considerable number of missing values. However, upon the absence of relevant structured data to learn and compute repairs, we can resort to textual data such as that presented in the bottom of Fig. 1. The sample text can be used to enrich the integrated data by filling in the missing values (e.g., '*Complications*' related to '*Acoustic Neuroma*') but also to complete partial data (e.g., '*Anatomies*' related to '*Acoustic Neuroma*').

The idea of enhancing data cleaning methods with external information has already been adopted in systems such as KATARA [16], using knowledge bases and crowdsourcing, or Cleenex [17], that leverages user feedback to define an iterative process. Yet, to the best of our knowledge no missing value imputation solution exists that exploits raw textual data (i.e., without annotations). On the one hand, the data management community has proposed several information extraction techniques [18], aiming to extract relational views from textual sources. However, the generation of structured views from textual data is limited to those extraction rules that can be defined over the text [19]. On the other hand, the natural language processing (NLP) community, has proposed text conceptualization approaches [20], which aim to automatically label text portions with a label. State of the art techniques for text conceptualization include (a) *Transformer-based Language Models* (LMs) to perform Entity Recognition (ER) from text, and (b) zero-shot approaches with *Large Language Models* (LLMs). However, the former require domain-specific large annotated data with rich context [21]. At the same time, they also need to be re-annotated and re-trained when the reference schema changes, while they also suffer from bias towards the most frequent entity, inconsistent performance, and demand vast resources to be trained (see Section II). In such a data integration setting, LMs perform poorly when trained with structured data only [22], as we experimentally validate in Section VI. The latter approaches, despite their massive popularity, often fail due to their uncertainty and tendency to hallucinate [23], [24]. LLMs struggle to maintain recall and precision when prompted with large text and complex concept categories [25]. Because of their attention mechanism, they often overlook fine-grained instances in the text, which are crucial for IE (see for instance a prompt to *ChatGPT*[1]). Furthermore, resource-wise, it is not feasible to train or fine-tune an LLM like GPT-4 on a case-by-case basis, a situation that often pertains to organizations typically dealing with their own data for integration purposes [26].

Our Proposal. In sight of the limitations of the current approaches, we present THOR[2] (Text Homogenization from Oblivion to Reality), a novel approach that formulates the data sparsity problem for qualitative data as an **Entity-Centric Slot-Filling task** [27], [28]. This is a text conceptualization task that involves extracting specific entities from large text corpora and mapping them into predefined concept categories. THOR uses the integrated schema concepts as the predefined concepts for the slot-filling task, and the structured data instances as patterns to identify entities from text. Specifically, THOR uses syntactic and semantic textual similarity measures to decompose sentences into chunks and identify entities. Then, entities are labeled with regard to the reference schema concepts via a fine-tuned similarity matcher. Unlike LMs, THOR only uses the non-contextual structured data sources and requires no further training. Thus, the cost-effort of our approach is dramatically smaller both in the training effort, since we bypass the need for large-scale annotated data, and the computational cost. Further, it easily adapts when the reference data integration schema evolves and adapts to the dynamic and evolving data integration scenario. As result, THOR's proposal lies in between two worlds, since it applies NLP techniques over structured data (considering both schema and instances, which lead the process) and textual data.

[1]**`ChatGPT Demo:`** https://bit.ly/ChatGPT-Demo
[2]**`Code & Artifacts`**: https://github.com/dtim-upc/THOR

The **contributions** of this paper are summarized as follows:

- We introduce a lightweight yet powerful approach for mitigating data sparsity by slot-filling the integrated data using conceptualized entities from external texts.
- While traditional imputation methods predominantly concentrate on categorical, numeric, or quantitative data; our contribution is focused on mitigating multi-valued qualitative data sparsity by extracting dispersed information regarding a specific entity from multiple text documents (entity-centric), enhancing the overall quality and completeness of integrated data.
- Our method significantly outperforms state-of-the-art (Large) Language Models in text conceptualization, particularly in entity recognition tasks, for data integration settings. Additionally, it offers the flexibility to be tuned for either precision or recall, based on user preferences.
- Compared to methods that necessitate the human annotation process, our approach showcases substantial reductions in time, resource consumption, and effort, highlighting its practicality for real-world applications.

II. Related Work

Recent advances in data science have enabled novel data integration techniques that extract and integrate information from diverse data sources on a large scale [29]. Despite being a well-researched area in data engineering, the task of mitigating incomplete data during data integration is often overlooked due to its inherent challenges [30]. To tackle this problem, we can apply existing techniques to extract, organize and enrich [31] structured information from unstructured or semi-structured data. We identify three trends that suit this paper's objective to mitigate data sparsity for integrated data: statistical data imputation, entity-recognition techniques, and slot-filling.

A. Statistical Data Imputation

Most conventional techniques [32] to impute data rely on statistical measures such as mean substitution, frequent category (mode) imputation, and maximum likelihood estimates; they are predominantly focused on either categorical, numeric, or quantitative data [33], [34]. Although most techniques perform quite well for numeric data [35], [36], their performance decreases for qualitative data [37]. From a statistical point of view, data imputation in data integration scenarios is more challenging than data imputation in single-database scenarios [30], since traditional statistical techniques do not deal with the *heteroscedasticity* (i.e., variance inconsistency) and *non-independent and identically distributed* (i.e., Non-IID) nature of data integrated from different sources [38]. As a consequence, a common practice is to simply ignore missing values [39]. Methods that try to mitigate qualitative missing values are either too limited in the sense that they only consider a fixed number of variables [40] or heavily dependent on domain-specific large datasets [37].

B. *Entity Recognition-based techniques*

The problem of extracting information from textual data has been deeply studied in the IE and NLP communities. IE techniques encompass several tasks, being one of them *Entity Recognition* (ER) [41] [42], which intends to identify entities mentioned in natural language text into pre-defined categories. Yet, traditional ER techniques involve creating lexicons or dictionaries containing a list of entities with specific tagging rules [43]. Although these lexicon and rule-based approaches are useful on small domains, they often tend to fall short in scenarios having complex entity types. Developing those rules also requires strong domain knowledge; hence, these systems are not generic. Alternatively, machine learning (ML) methods, such as Support Vector Machines (SVM), Decision Trees, Hidden Markov Models (HMM) [44], and Conditional Random Fields (CRFs) [45]–[47], have also been employed for this task. These are trained in a supervised fashion using annotated text samples with entity labels and ad-hoc feature engineering. However, since these models solely rely on training examples, they struggle to generalize to new, unseen data.

Most advanced state-of-the-art ER techniques use *Neural Language Models* based on Transformer architectures with Attention mechanisms (i.e., BERT, XLNet, RoBERTa, T5, GPT, LlaMA, UniversalNER). LMs exploit contextual information (i.e., neighboring words) and distributional semantics (words with similar distributions have similar meanings) [48] via an unsupervised pre-training phase. Then, they apply a supervised strategy called Masked Language Modeling (MLM) [49]–[51]. During MLM training, it uses a placeholder or mask, to hide the entity labels in the annotated data and predict the masked labels from the neighboring words. These techniques, in the presence of a large corpus of annotated text, outperforms previous ER techniques [25], [52]–[56].

Regardless of the technique used, we find two main approaches when predicting the entity label in an ER technique. Either identify the label by only considering the text at hand or the text and a reference schema. The former is a common practice in the knowledge base (KB) population and enrichment [57]–[62], where they extract information from text in the form of *subject-verb-object* relations and use entity linking [63] techniques to enrich the KB with the extracted subjects and objects [64]–[69]. These techniques rely on syntactic categories (e.g., noun phrases) and tend to generate long-tail entities that require further processing before being used [70]. For this reason, the current trend is to conceptualize entities in the textual data following a reference schema [71]. State-of-the-art techniques [72]–[74] can be used in this regard and convert the information embedded in the text into a structured representation via conceptualization. This is also the chosen approach to facilitate the enhancement [61], [75] and construction [76]–[80] of structured knowledge sources.

C. *Slot-Filling*

Slot-Filling [81] [82] [83] is a well-known technique that can be used to reduce data sparsity via conceptualized text. Slot filling involves populating entity-specific templates (e.g., *[Acoustic Neuroma, causes, <slot>]*) with information extracted from text. It can be either *document-centric*, focusing on entities represented by a single document, or *entity-centric*, where information about a concept is spread across multiple documents in a corpus. Most advanced slot-filling techniques, essentially, follow the same principles as LM-based techniques for ER and suffer from the same problems: (i) the need for a domain-specific large annotated data with rich context, (ii) the need to re-annotate and re-train when the reference schema changes, (iii) bias towards the most frequent entity type, (iv) inconsistent performance and hallucination and (v) resource-hungry training. Unfortunately, these assumptions do not hold in data integration. Although there is available data to work on, these are primarily structured with limited context (e.g. tabular data), making it difficult for LMs to work properly. Also, despite organizations have lots of textual data, those are not annotated. In the absence of these requirements, the performance of LMs quickly decreases [84]. Further, in a data integration system, the integrated schema evolves, which would require re-annotating the corpus and re-training the model. Indeed, adapting these methods in the presence of structured data, which by definition have limited context, is nowadays an open research area [85]–[87]. Alternatively, zero-shot Large Language Models such as GPT [88], despite their massive popularity, are unsuited here due to their unpredictable nature [23], inconsistent performance [24] and tendency to hallucinate [89]. LLMs also struggle to maintain recall and precision with large text corpora in the presence of complex concept categories [25]. Furthermore, it is not feasible in terms of resources (i.e., compute power and time) to train or fine-tune a large model like GPT on a case-by-case basis [26].

D. *Research Gap*

In order to overcome the limitation of the related work to mitigate incomplete data over an integrated system using textual data, THOR proposes a novel *Entity-Centric Slot-Filling* strategy where incomplete information is extracted via conceptualization with regard to the concepts defined within the data integration system. THOR's distinguishing feature is its lightweight (resources-wise) approach that does not require any annotated text for training; instead, it leverages existing structured data to reduce data sparsity for qualitative data. In these settings, THOR outperforms both traditional and advanced LM-based approaches, as shown in Section V.

III. Problem Statement

In this section, we introduce the formal background of our approach and state the problem definition. Table I summarizes the notation used throughout the paper.

A. *Preliminaries*

Concept. A *concept* C represents an idea, category, or class of things. A concept has *instances*. For example, '*Tuberculosis*' and '*Acne*' are instances of the '*Disease*' concept. We denote a concept instance as c, and use $dom(C)$ to denote the domain of concept C (i.e., the set of all possible instances of C).

TABLE I
NOTATION'S USED IN OUR PAPER

Symbol	Meaning
C; C^*	a concept; the subject concept
c; c^*	an instance; a subject
$\mathcal{C}$	the set of concepts acting as a schema
R; $R.C$	a table; the column corresponding to concept C
r; $r.C$	a row; the values (instances) in column C
s	a sentence
p	a phrase
e; $e.p$; $e.C$	an entity; the entity phrase; the entity concept

Table. We consider a concept-oriented *schema*, defined as a collection of concepts $\mathcal{C}$, among which one concept, termed the *subject concept* $C^* \in \mathcal{C}$ plays the role of the primary key. We assume a relational *table* R that adheres to the schema $\mathcal{C}$. Since the columns of R correspond to the concepts $\mathcal{C}$, we use the terms column and concept interchangeably.

We denote a *row* of table R as r, and use $r.C$ to refer to the value(s) of the row for concept $C \in \mathcal{C}$. Every row has a single value for the subject concept, while it can be multi-valued for the other concepts. We denote as $R.C$ the set of all values in column C of table R; these values are concept instances, i.e., belong to $dom(C)$.

Document. A *document* D is a collection of *sentences*. A sentence s is a sequence of *words*. A *phrase* p is a subsequence of a sentence. A *noun phrase* is a special phrase that contains a noun (or a pronoun) and associated modifiers (e.g., adjectives, determiners, or other qualifying words). Noun phrases often serve as subjects, objects, or complements within a sentence. For example, in sentence '*Tuberculosis generally damages the lungs*', '*Tuberculosis*' and '*the lungs*' are noun phrases.

Entity. A conceptualized entity, or simply an *entity* e is a phrase that is associated with a concept, i.e., a pair $\langle p, C\rangle$, where $p \in dom(C)$. For instance, $\langle$'*Tuberculosis*', '*Disease*'$\rangle$ is an entity extracted from the sentence '*Tuberculosis generally damages the lungs*'. Entities typically consist of noun phrases (or subsequences thereof).

B. Problem Definition

Problem 1: Given a table R that follows the concept-oriented schema $\mathcal{C}$, and a document D, the goal is to identify entities from D and use them to enrich R.

IV. THE THOR PIPELINE

In this section, we describe the THOR pipeline, which addresses Problem 1. Specifically, it receives as input a table R with its concept-oriented schema $\mathcal{C}$, and a document D, and outputs an enriched version of R that is populated with entities derived from D.

Figure 2 presents a visual depiction of the THOR pipeline, while Algorithm 1 describes it in pseudocode. The THOR pipeline consists of three phases, ① Preparation, ② Entity Extraction, and ③ Slot Filling, which are described in the following subsections. Briefly, Phase ① performs two tasks, it (a) segments the document into sentences and determines the subject instance per sentence, and (b) fine-tunes a semantic matcher based on the concepts and their instances. Phase ② extracts (conceptualized) entities from the document D considering both the semantic and syntactic similarity of entities to concept instances. Phase ③ uses the extracted entities to fill slots in (i.e., enrich) table R.

Algorithm 1: THOR

```
Input: Table $R$, schema $\mathcal{C}$, document $D$, threshold $\tau$
Output: Enriched Table $R'$

① Preparation
1  $\{\langle c^*, s\rangle\} \leftarrow$ SEGMENT($D$, $R.C^*$)
2  MATCHER.FINETUNE($\mathcal{C}$, $R$, $\tau$)

② Entity Extraction
3  foreach $c^* \in R.C^*$ do
4      $E[c^*] \leftarrow \varnothing$            ▷ Initialize extracted entities per subject instance
5  foreach $\langle c^*, s\rangle \in \{\langle c^*, s\rangle\}$ do
6      $\{p\} \leftarrow$ PARSER.EXTRACTNOUNPHRASES(s)
7      foreach $p \in \{p\}$ do
8          $\{(e, c_m)\} \leftarrow$ MATCHER.MATCH($p$)        ▷ Candidate entities for $p$
9          foreach $(e, c_m) \in \{(e, c_m)\}$ do
10             $e.\text{score}_s \leftarrow$ MATCHER.SIMILARITY($e.p$, $c_m$)
11             $e.\text{score}_w \leftarrow$ JACCARD($e.p$, $c_m$)
12             $e.\text{score}_c \leftarrow$ GESTALT($e.p$, $c_m$)
13             $e.\text{score} \leftarrow (e.\text{score}_s + e.\text{score}_w + e.\text{score}_c)/3$
14         $e_{\text{best}} \leftarrow \arg\max_{\{e\}}\{e.\text{score}\}$        ▷ Best candidate entity for $p$
15         $E[c^*] \leftarrow E[c^*] \cup \{e_{\text{best}}\}$

③ Slot Filling
16 foreach $c^* \in R.C^*$ do
17     $r \leftarrow \text{SELECTION}_{C^*=c^*}(R)$                ▷ Select row with key $c^*$
18     foreach $e \in E[c^*]$ do                 ▷ For each entity related to $c^*$
19         $C \leftarrow e.C$                    ▷ Get concept associated with the entity
20         $r.C \leftarrow r.C \cup \{e.p\}$         ▷ Add phrase $e.p$ to column $C$ of $r$
```

A. Preparation

Phase ① (lines 1–2 in Algorithm 1) involves the segmentation of the document and fine-tuning of the semantic matcher.

Document Segmentation. The goal of segmentation is to split the given document into sentences and associate each sentence with an instance of the subject concept (or with none if the sentence is not related to the subject concept). In our running example (Figure 1), '*Disease*' is the subject concept, and the document contains three sentences, among which the first two relate to instance '*Acoustic Neuroma*', and the last to instance '*Tuberculosis*'. Segmentation is typically an easy task, as paragraphs, or even entire documents, often talk about a specific subject instance. If that is not the case, we employ semantic matching-based techniques (see Section IV-B) to classify a sentence into one of the predefined subject instances. Line 1 in Algorithm 1 corresponds to the segmentation task, and shows that the input is the document D and the set $R.C^*$ of all subject instances, and that the output is a set of $\langle$subject instance, sentence$\rangle$ pairs, represented as $\langle c^*, s\rangle$.

Fine-Tuning a Semantic Matcher. *Semantics* refers to the meaning of natural language expressions such as words, phrases, and sentences, [90], [91]. *Semantic similarity matching* is the NLP task of deciding how similar the meaning of two phrases are. THOR applies semantic similarity matching to identify candidate entities that can enrich a given table. Specifically, THOR employs a generic semantic similarity matcher[3] built on top of the popular spaCy NLP library, and

[3]https://github.com/gandersen101/spaczz#SimilarityMatcher

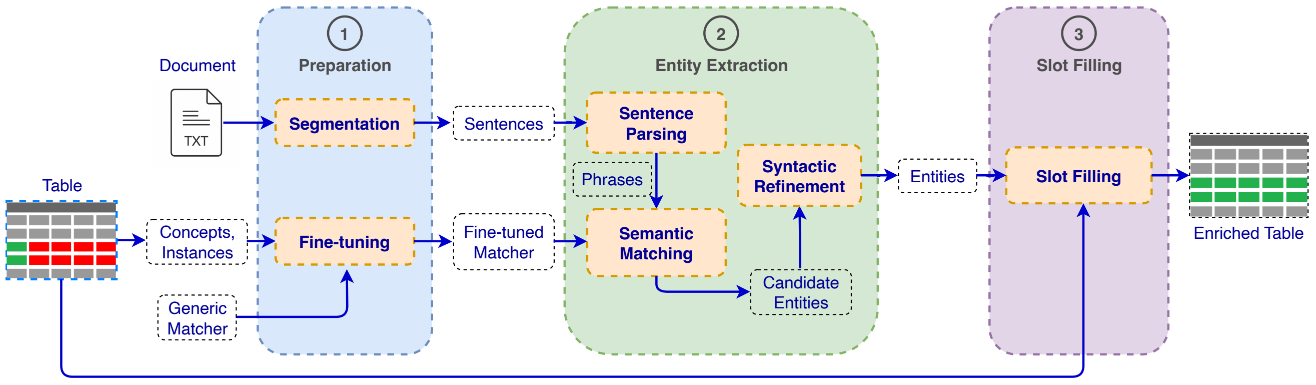


Fig. 2. The THOR Pipeline.

fine-tunes it for the domain at hand, as represented by the concepts and their instances.

Internally, the matcher uses pre-trained word embeddings [92], hereafter simply called *vectors*, for the English language (trained on the *OntoNotes 5.0* [93] and *Wikipedia* [94] corpora) to determine the semantic similarity of two given words. The goal of fine-tuning the matcher is to associate each concept C with a set of *representative vectors* that *semantically cover* the domain of C, in the sense that the vector of an unknown instance of C should be similar to one of these representative vectors. To achieve this for C, we include as representative vectors for C the embeddings of the known instances of C (i.e., those in $R.C$), acting as *seeds*, and also include other vectors that are highly similar to the seeds (i.e., beyond a user-defined threshold τ). Together they form the *representative instances* that includes both known instances and unknown instances. By setting the τ, we can choose to be more restrictive (precision-oriented) or more inclusive (recall-oriented).

This fine-tuning process, depicted in Line 2 of Algorithm 1, allows THOR to achieve domain adaptation via *weak supervision* as we do not use any annotated training data, in contrast to other approaches (e.g., [67], [68]) that require either a large set of rules or a large domain-specific training dataset. The following table illustrates the fine-tuning process, using the concepts and instances in the table of Figure 1. For the concept '*Anatomy*', a known instance is '*nervous system*' (shown underlined), which acting as a seed returns additional representative words, such as '*brain*' and '*nerve*', that are associated with '*Anatomy*'. Similarly, the table depicts the seed and other representative words associated with '*Complication*'.

Concept	**Representative Instances**
'*Anatomy*'	{. . . , '<u>*nervous system*</u>', '*brain*', '*nerve*', . . . }
'*Complication*'	{. . . , '<u>*skin cancer*</u>', '*cancer*', '*tumor*', . . . }

B. Entity Extraction

Phase $\textcircled{2}$ (lines 3–15 in Algorithm 1) considers each subject instance, sentence pair $\langle c^*, s\rangle$ resulting from the document segmentation of the previous phase, and produces a set of entities extracted from s. These are stored in a map structure $E[c^*]$ that associates a subject instance c^* with all related extracted entities, and which is initialized in lines 3–4. The phase involves parsing the sentence into phrases, and applying semantic matching and syntactic refinement over them.

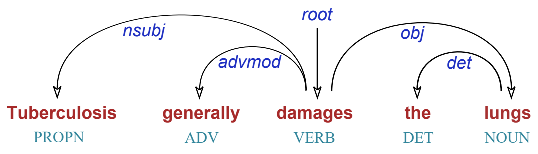


Fig. 3. Example of a dependency parse tree.

Sentence Parsing. To identify relevant entities from the text, THOR first exploits linguistic properties. Specifically, it analyzes the structure of the sentence in terms of the grammatical relations among words and phrases, a process known as *dependency parsing* [76]. Figure 3 depicts the parser's output given an input sentence from our running example. The parser yields a *tree* consisting of directed edges between words. Each *edge* represents the *thematic role* between two words. Thus, the parser produces a set of binary relations. For example, the root verb '*damages*' is related to the word '*Tuberculosis*' having the role *nominal subject* (nsubj) and with the word '*lungs*' through the *object* (obj) thematic role. Thematic roles can help determine *subject-verb-object* relations in a sentence [68]. In addition to thematic roles, the parser performs *part-of-speech tagging*, associating with each word their grammatical function (e.g., VERB, ADJECTIVE, NOUN).

THOR uses the dependency parse tree to extract *noun phrases*. A noun phrase is a subtree that has at its root a noun (NOUN), pronoun (PRON), or proper noun (PROPN), and might also include leading or trailing modifiers, such as adjectives (ADJ) and determiners (DET) [95]. For instance, in Figure 3, '*the lungs*' is a noun phrase having the noun '*lungs*' at its root. THOR strips from noun phrases any leading or trailing stop-words (such as *a*, *of*, *the*). Given a sentence s, the parser outputs a set $\{p\}$ of phrases, as indicated in line 6 of Algorithm 1. Continuing the running example, THOR will generate the set of phrases {'*Tuberculosis*', '*lungs*'} from the sentence depicted in Figure 3.

Semantic Matching and Syntactic Refinement. Once the phrases have been generated, THOR iterates over each phrase p (lines 7–15). Given p, THOR performs (a) *semantic matching*, i.e., extracts candidate entities from p (line 8), and

(b) *syntactic refinement*, i.e., assesses entities with different criteria (lines 9–13) and selects the best (lines 14–15).

Semantic Matching. The output of semantic matching is a collection of candidate entities $\{e\}$ extracted from p. A candidate entity e concerns a subphrase of p, denoted as $e.p$, and is associated with a concept, denoted as $e.C$. Recall from Section IV-A that the matcher is fine-tuned so that each concept is associated with a set of representative vectors (which are the embeddings of instances from the table and similar words from the vectors). Essentially, the collection of representative vectors of a concept C acts as a cluster.

Semantic matching, depicted in line 8 of Algorithm 1, works as follows. The matcher generates all subphrases from p. Each subphrase, denoted as $e.p$, is embedded and is represented as a *query vector*. By computing the mean pairwise similarity between the query vector and the representative vectors of each concept/cluster, the matcher identifies the concept $e.C$ that semantically best fits the subphrase $e.p$. In addition, the matcher determines the concept instance c_m of $e.C$ that has the highest semantic similarity to $e.p$; this matched instance c_m is later used to assess the quality of the extracted entity.

As an example, consider the noun phrase '*slow-growing non-cancerous brain tumor*', from the document in Figure 1. Semantic matching generates all its subphrases and investigates if they can form entities, i.e., if they are highly similar to representative words of a concept. Suppose, the two entities shown in the following table are recognized. Subphrase '*brain*' is mapped to concept '*Anatomy*' due to its semantic similarity with seed instance '*nervous system*'. Similarly, '*non-cancerous brain tumor*' is matched to '*Complication*' via '*skin cancer*'.

Entity	**Phrase** ($e.p$)	**Concept** ($e.C$)	**Seed Instance** (c_m)
e_1	'*brain*'	'*Anatomy*'	'*nervous system*'
e_2	'*non-cancerous brain tumor*'	'*Complication*'	'*skin cancer*'

Syntactic Refinement. Semantic similarity matching can identify novel instances (e.g., '*Malaria*') not seen among the known instances of a concept (e.g., '*Disease*'). However, when concepts have instances that are similar across concepts (e.g., 'blood' is an '*Anatomy*', while 'blood clot' is a '*Complication*'), the semantic matcher might extract entities for a given phrase (e.g., 'blood vessels') that match both concepts. To mitigate this issue, THOR further refines entities based on syntactic criteria.

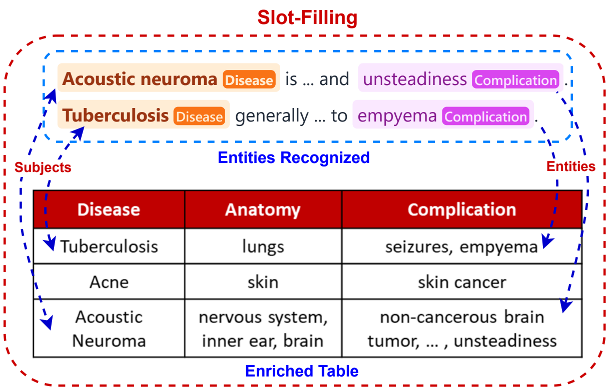
Recall that for each candidate entity e, the semantic matcher determines its best matching concept instance c_m. THOR computes three scores to assess the quality of an entity e. The first, denoted as $e.\text{score}_s$ in line 10, is the semantic similarity between the entity (more precisely, the phrase $e.p$) and its matched instance. The second, denoted as $e.\text{score}_w$ in line 11, is a word-level syntactic similarity between the entity and its matched instance; specifically, phrases $e.p$ and c_m are viewed as a set of words, and THOR computes the Jaccard (aka intersection over union) similarity between the sets. The third, denoted as $e.\text{score}_c$ in line 12, is also a syntactic similarity between the entity and its matched instance but is computed at the character level; specifically, $e.p$ and c_m are viewed as strings and the gestalt pattern matching algorithm [96] is used to compute a string similarity score. All three scores take values in the $[0, 1]$ range, where higher values mean higher similarity. THOR combines them into an average score, and selects the highest scoring entity e_{best} as the best entity extracted from phrase p. This entity is added to the list of other entities associated with the subject concept c^* of the sentence that phrase p belongs to (line 15).

Continuing the running example, where the two candidate entities e_1 and e_2 have been discovered, THOR next computes the three scores and averages them. Observe that the phrase '*non-cancerous brain tumor*' of e_2 has a higher syntactic similarity to its seed instance than the phrase of e_1 has. Therefore, the average score for e_2 is higher, and this entity is selected as the best entity discovered from noun phrase '*slow-growing non-cancerous brain tumor*'.

Entity	$e.\text{score}_m$	$e.\text{score}_w$	$e.\text{score}_c$	$e.\text{score}$
e_1	1	0	0.45	0.48
e_2	0.8	0.4	0.39	**0.53**

C. Slot Filling

Phase ③ (lines 16–20 in Algorithm 1) starts once the document has been conceptualized, i.e., conceptualized entities have been extracted, and involves enriching the table R. THOR iterates over subject instances, and for each subject instance c^*, the row r that has value c^* in each subject column C^* is selected (line 17). Then, for every entity e related to subject c^*, THOR fills in the slot that corresponds to row r and column $e.C$ with the extracted phrase $e.p$ (lines 18–20).



Slot-Filling
Acoustic neuroma Disease is ... and unsteadiness Complication .
Tuberculosis Disease generally ... to empyema Complication .
Entities Recognized
Subjects
Entities
Enriched Table

Disease	Anatomy	Complication
Tuberculosis	lungs	seizures, empyema
Acne	skin	skin cancer
Acoustic Neuroma	nervous system, inner ear, brain	non-cancerous brain tumor, ... , unsteadiness

Fig. 4. The slot-filling phase on our Running Example that results in enriching the table with the extracted conceptualized entities.

Figure 4 illustrates slot filling, where two entities, '*unsteadiness*' and '*empyema*', related to two subjects, '*Acoustic neuroma*' and '*Tuberculosis*', respectively, fill in two slots for the concept '*Complication*'. Similarly, following the running example, we will also have '*non-cancerous brain tumor*' assigned to the slot of '*Complication*' for '*Acoustic neuroma*'.

V. Experimental Settings

In our experiments, we aim at evaluating the entity-centric slot-filling task defined in our problem statement. Given a table R and a set of documents $\mathcal{D}$, the goal is to extract entities from $\mathcal{D}$ and use them to enrich R. To that end, we use two different datasets, and evaluate the slot-filling task on THOR and SOTA competitors by means of three experiments.

A. Datasets

Most available benchmarks for slot-filling tasks [97]–[102] are (a) either too narrow and specific (with only few concepts), or (b) annotated with a standard predefined tag-set (e.g., location, organization), or (c) contains a small number of average tokens per document, deeming them insufficient. We instead consider two datasets with a wide variety of entity types, as found in real-life settings. First, we consider a health-related data scenario that contains **Disease A-Z** information with **Conditions**. The second dataset consists of **Résumé**(s) containing job-seekers information.

TABLE II
Concepts in the Disease A-Z and Résumé Datasets (in bold we highlight the subject concept)

Dataset	Concepts
Disease A-Z	**Disease** (C^*), Anatomy, Cause, Complication, Composition, Diagnosis, Medicine, Precaution, Riskfactor, Surgery, Symptom
Résumé	**Name** (C^*), Awards, Certification, Degree, University, College Name, Language, Location, Worked As, Skills, Companies Worked At, Years Of Experience

Structured data sources. Each dataset contains a table R. These tables were created by combining different available online data. The concepts defining the schema ($\mathcal{C}$) of the tables (R) used in both datasets are described in Table II. The *Disease A-Z* table (R_D), created from 10 sources, contains 11 concepts, 284 instances of the subject concept (i.e., rows), and a total of 4,706 instances. The *Résumé* table (R_R), created from 12 sources, contains 12 concepts, 201 instances of subject concept (i.e., rows), and a total of 3,119 instances. Both tables can be found in our Github[4].

Text data sources. Additionally, each dataset contains a set of documents ($\mathcal{D}$) with which we seek to enrich the tables. To create the ground truth $\mathcal{D}_D$ for the *Disease A-Z* dataset, we collected information related to *Disease/Conditions A-Z*[5] from several major health portals such as WHO, NHS, and CDC; ensuring information trustworthiness. We manually collected texts on 314 diseases and each disease was stored in a different document. The Résumé dataset $\mathcal{D}_R$ was generated by collecting information related to ***Résumé*** [6] of job-seekers [103] and generated 54 documents (each containing 5 CVs). Table III, provides an overview of the annotated text dataset statistics, as well as the generated splits. The annotations of the identified entities were made with regard to the reference tables schema (i.e., $\mathcal{C}$) described in Table II. The description of the annotation process is left out of this paper due to space limitations. However, we report it along with the artifacts provided[4].

TABLE III
Annotated Text Sources Statistics: Disease A-Z and Résumé

#	Disease A-Z			Résumé		
	Train	Valid.	Test	Train	Valid.	Test
|dom(C^*)|	240	61	13	100	70	100
Documents	1438	366	90	20	14	20
Entities	18539	3989	2222	1656	1463	2140
Words	168816	38722	19237	41675	25389	38459

Evaluation strategy. For each dataset, we split the ground truth text data into three sets: train $\mathcal{D}_{train}$, validation $\mathcal{D}_{val}$ and test $\mathcal{D}_{test}$, following standard practices in slot-filling tasks, as described in Table III. We also created two ***test tables*** ($R_{D_{test}}$ and $R_{R_{test}}$) from the ground truth test set $\mathcal{D}_{test}$. During the evaluation, we deleted the instances of all concepts from these test tables ($R_{D_{test'}}$ and $R_{R_{test'}}$) except for the subject concepts (i.e., '*Disease*' and '*Name*', respectively). Then, we evaluate THOR and its SOTA competitors by extracting entities from the test set $\mathcal{D}_{test}$ and slot-filling $R_{D_{test'}}$ and $R_{R_{test'}}$. This setting represents the worst-case scenario for the entity-centric slot-filling task at hand. The evaluation metrics used are discussed below, and they are computed by comparing the slot-filled values in $R_{D_{test'}}$ and $R_{R_{test'}}$ with $R_{D_{test}}$ and $R_{R_{test}}$.

Evaluation Metrics. We opted for a well-known metric for text conceptualization, proposed during SemEval-2013[7] [104], based on ***Precision*** (P), ***Recall*** (R), and ***F1-score*** that considers partial matching. This is important for evaluating any IE system [105], especially for the implicit conceptualization tasks, since an entity could be tagged differently than the ground truth yet still be partially correct. Take, for example, the entity '*main (vestibular) nerve*' where a system might identify only the '*vestibular*' portion as an '*Anatomy*'. Although this prediction is correct to an extent, a stricter metric commonly applied in classification tasks would consider it wrong. Therefore, selecting a representative evaluation metric is crucial. Another metric we will be using is the ***Sensitivity*** to have a deeper understanding of the models capabilities to recognize the entities in the ground truth (i.e., true positives) for each of the concept categories. This is vital in order to understand the direct effect of missed predictions (false negatives) by considering the partial results as well.

B. Experiment-1: Comparison Against State-Of-The-Art

In this experiment, we evaluate and compare the performance of THOR in the presence of complex real-world concepts against the SOTA methods (see Section II) using the *Disease A-Z* dataset. Table IV gives a compact overview of these approaches, including our own.

[4]**Datasets:** https://github.com/dtim-upc/THOR/tree/main/Dataset
[5]**Disease A-Z Data Source:** https://www.nhs.uk/conditions/acne/
[6]**Résumé Data Source:** https://www.romanshilpakar.com.np/blog/6
[7]**SemEval-2013 Evaluation:** https://github.com/MantisAI/nervaluate

TABLE IV
OVERVIEW OF THE METHODS IN OUR EXPERIMENTS.

Acronym	Algorithm/Model	Embedding/Pre-training Data	Fine-tuning Data
`Baseline`	Aho–Corasick	–	Structured Data (R_D,R_R)
`LM-SD`	RoBERTa-Base	BookCorpus, CC-News, OntoNotes5, OpenWebText, ClearNLP, WordNet 3.0, Stories, English Wikipedia	Structured Data (R_D,R_R)
`LM-Human`	RoBERTa-Base	Same as LM-SD	Annotated Text (D_{train})
`GPT-4`	GPT-4	Part of WWW (∼13 trillion tokens), Human feedback	–
`UniNER`	GPT-3.5	Pile-NER, 40 other NER datasets	–
`THOR`	Novel Hybrid	Static embeddings from OntoNotes5, Explosion, WordNet 3.0	Structured Data (R_D,R_R)

Baseline. A traditional ER method that uses substring-search for exact syntactic matching (Aho–Corasick algorithm [106]). This technique does not require any complex training and represents a fair baseline to compare with. It uses structured data as patterns to build a dictionary or lexicon, which is then further used to match all sub-strings from the text.

LM Fine-tuned on Structured Data (LM-SD). A pre-trained standard transformer-based Language Model fine-tuned for the specific task of ER. For this task-specific fine-tuning of **LM-SD**, we used the structured data sources. The choice of LM was a pre-trained RoBERTa-Base [50] model based on dynamic masking at training time, which showed state-of-the-art performance for ER [53]. We also verified this specific model's performance by comparing it with 6 popular LMs used in ER literature [107] [108]: BERT-base (cased and uncased), DistilBERT-base (uncased), BioBERT, ELECTRA-base-NER, vanilla RoBERTa, XLM-RoBERTa-base. The model we chose was pre-trained with 7 well-known corpora: BookCorpus, CC-News, OntoNotes5, OpenWebText, ClearNLP, WordNet 3.0, Stories, and English Wikipedia—making it a very strong candidate for our evaluation. On top of additional data and more extensive pre-training, LM uses contextualized word embeddings, which makes it a very powerful tool for capturing semantics in text. Since THOR uses only structured data, we wanted to be fair regarding fine-tuning. Thus, this alternative will show how good a standard LM is, compared to our approach when they do not have any annotated text available.

SOTA LLMs (GPT-4 and UnivNER). Recently, prompt-based zero-shot Large Language Models have been the go-to option when annotated data are unavailable. Thus two prompt-based zero-shot SOTA LLMs (GPT-4 [88] and UniversalNER [25]) are used without further fine-tuning. This setting will provide a comparison of how well these models perform when we have a complex schema. We chose two models to capture two facets of this recent development: (a) the latest version of OpenAI's GPT-4 [88], which has been trained with around 13 trillion tokens from the Internet, meaning it is the SOTA option for generalizability when it comes to natural language task [109] [110], and (b) UniversalNER [25] model that showed SOTA results on nearly all the popular benchmark ER datasets—making it the strongest competitor when it comes to the conceptualization task. Since it is not possible to fine-tune these models for the specific task of conceptualization due to the high computation requirements, we used well-defined prompts[8] for GPT-4 and the recommended prompt[9] by the author of UniversalNER. The context window plays an important role for these models, as it defines the number of tokens taken as input when generating responses. We thus used the maximum possible context window supported by each model (16k for GPT-4 and 3k for UniversalNER).

[8]**GPT-4:** https://github.com/dtim-upc/THOR/blob/main/GPT-4/prompts.txt
[9]**UniversalNER:** https://huggingface.co/Universal-NER/UniNER-7B-all

LM with Human Annotated Text (LM-Human). A pre-trained LM (RoBERTa) fine-tuned with properly contextualized text data (i.e., D_{train}) of each dataset. This represents the ideal situation for LMs. This way, we can compare our approach to the best possible LMs scenario. It uses the same RoBERTa-Base model with settings similar to LM-SD for pre-training. However, since language models are well-equipped to handle contextualized text data and perform best when supplied with task-specific (in this case ER) extensively labeled text, we aim to compare THOR with this best-case scenario for LM when fine-tuned with properly annotated text with concepts as labels. In a practical data integration scenario, it is nearly impossible to annotate a massive amount of data. Nevertheless, we compare our weakly supervised approach with this hypothetical case in order to assess the performance gains from manual annotation for LM, keeping in mind the need to re-annotate and re-train every time the reference schema changes.

C. *Experiment-2: Manual Vs. Automatic Annotation*

LMs trained with an annotated large corpus are the best option to perform entity-centric slot-filling. However, they come with several costs that make this scenario not feasible in realistic data integration settings. As discussed in Section II, we scrutinize the amount of annotated data required to make LMs outperform our approach and the computational resources required for the fine-tuning when executing Experiment-1. In this experiment, we report on:

1) **Performance Comparison with regard to the amount of available annotated data:** Dividing the annotated text data into different sizes to evaluate the performance of the LM when fine-tuned with each. We aim to identify the amount of annotated data required to outperform our approach.
2) **Time and Effort Analysis:** Based on the previous analysis, we measure and compare timescales and efforts required in training an LM to reach the performance of our approach.

D. *Experiment-3: Generalizability*

In this experiment, we evaluate the generalizability of THOR. For that, we compare THOR with the SOTA approaches used in Experiment-1 (see Table IV) in a new domain (the **Résumé** dataset). We carefully checked that the Résumé dataset is not a part of the embedding or pre-training for any of the approaches, thus representing a common scenario in organizations where their data does not resemble that of available open datasets. Thus, in this experiment, we evaluate how all these approaches cope with mostly unseen and uncommon datasets.

VI. Experimental Evaluation and Results

We examine the experimental results based on the evaluation criteria defined in Section V. All artifacts associated to THOR as well as those related to the reproducibility of experimental results are openly available at this paper's companion website[10].

A. Experiment-1: Comparison Against State-Of-The-Art

We compare THOR against the SOTA techniques for ER on text conceptualization. Table V shows the overall results of THOR for different values of our similarity threshold (ranging from 0.5 to 1.0 with an interval of 0.1) in the upper columns. The lower part of the table lists the results of the comparative approaches. We considered both the fine-tuning time and inference time together. It shows our model improves on time as we go stricter on the value of the threshold τ. This is due to the fact that the stricter version of THOR (τ=1.0) produces few matching results compared to the more lenient one (τ=0.0). Thus, the syntactic matchers take way less time in order to rank the matching results. One other trend to see here is that, as we go stricter, THOR's precision (P) increases (from 0.39 to 0.63), providing the user with a way to be more precision-oriented. The recall (R), however, decreases due to the fact that it produces fewer results. We achieved our best overall F1 score of 0.56 at a threshold value of 0.70.

TABLE V
Comparative results for slot-filling on Disease A-Z Dataset

Exp.1	Model Name	Time(s)	P	R	F1
THOR	THOR (τ = 0.5)	1781	0.39	0.74	0.52
	THOR (τ = 0.6)	870	0.44	0.71	0.54
	THOR (τ = 0.7)	493	0.49	**0.64**	**0.56**
	THOR (τ = 0.8)	427	0.56	0.52	0.54
	THOR (τ = 0.9)	425	0.60	0.40	0.48
	THOR (τ = 1.0)	425	**0.63**	0.32	0.42
OTHER	Baseline	524	0.55	0.18	0.27
	LM-SD	3626	0.42	0.45	0.43
	GPT-4	-	0.49	0.38	0.43
	UniNER	3298	0.58	0.33	0.42
	LM-Human	3564	**0.83**	**0.56**	**0.66**

An in-depth comparison is depicted in the Precision-Recall curve of Fig. 5, which shows that THOR dominates all the models apart from LM-Human, which was trained using manually-labeled text. THOR also outperforms the competitors in terms of runtime, since inference time drops as the threshold increases (see Fig. 6). Going back to Fig. 5, the overall result shows THOR's superiority in comparison to both the SOTA LM (LM-SD) and LLMs (GPT-4, UnivNER). Although these are pre-trained with a massive amount of data (as discussed in Section V) in the case of LLMs, and further fine-tuned with structured data in the case of LM-SD, along with a peak on the properly labeled text from the validation set during fine-tuning in order to generalize, they tend to perform poorly as compared to THOR.

[10]https://github.com/dtim-upc/THOR

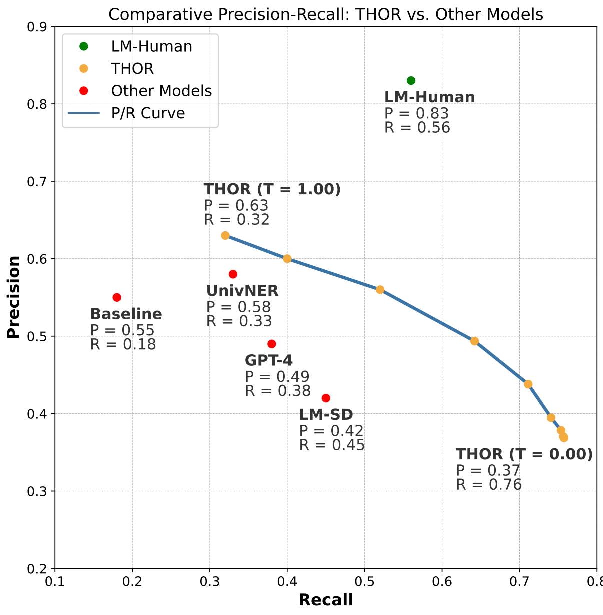


Fig. 5. Precision-Recall Curve for THOR (varying τ) and the alternatives

Next, we focus on the raw prediction counts made by the models (Table VI). Here, we compare our top-3 precision-oriented THOR models with the comparative approaches based on the ground truth vs. the predictions by these systems. From Table VI, it is clear that although LM-Human prevails in the overall F1 score, it does so by predicting less compared to other models such as THOR (τ=0.8) or LM-SD. LM-SD predicts more entities but produces a significantly higher number of *incorrect predictions* or False Positives (FP) compared to THOR. We can also observe that the Baseline predicts the lowest number since it tries to match exactly what it has in its dictionary. This is one of the reasons why the Baseline precision is high, but the recall is very low. This implies that traditional lexicon-based syntactic methods will fail in terms of detecting out-of-vocabulary (OOV) entities, thus ruling out exact matching techniques from the equation when it comes to enriching integrated data with such techniques. In contrast, THOR (τ=0.8) has the highest number $1,464$ of *correct predictions* or True Positives (TP), while being better compared to most of the models when we consider both of these parameters (TP and FP). This situation is further observed in Fig. 7, which shows the deviation from the ground truth in terms of *missed entities* or False Negatives (FN).

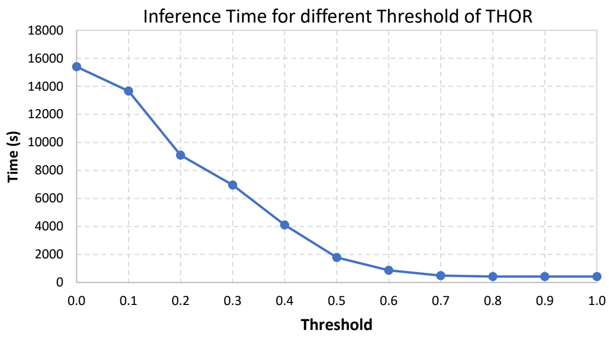


Fig. 6. Inference Time (seconds) of THOR for an Increasing Threshold Value

TABLE VI
THOR (Top-3 Precision) vs. Other Models: Comparative Results Based on Raw Prediction Count (TP, FP) on Disease A-Z Dataset

Exp.1	Model Name	Ground Truth Entities	Predicted Entities	Correct Predictions (TP)	Incorrect Predictions (FP)
	THOR (τ=0.8)		2069	**1464**	605
THOR	THOR (τ=0.9)	2222	1496	1129	367
	THOR (τ=1.0)		1123	886	237
	Baseline		725	**588**	137
	LM-SD		2421	1456	**965**
OTHER	GPT-4	2222	1724	1089	635
	UniNER		1272	951	321
	LM-Human		1494	1383	**111**

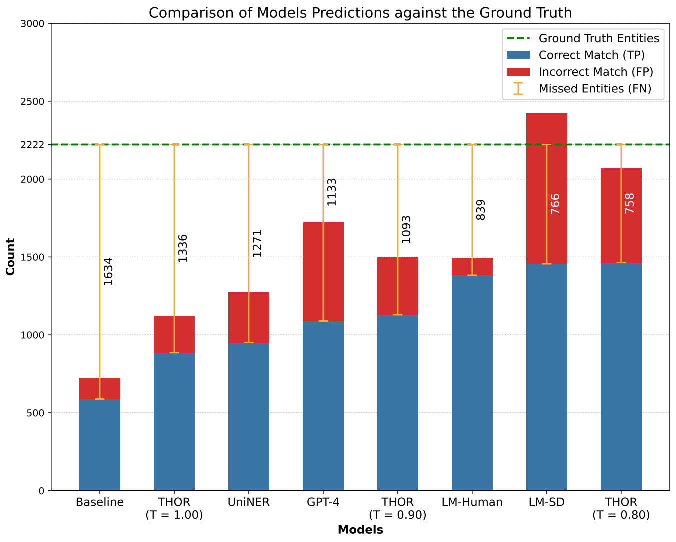


Fig. 7. Bar Chart Representing Overall Prediction Counts Based on TP, FP, and FN with respect to the Ground Truth Entities of Disease A-Z Dataset

In this experiment, we also observed the problem of **uncertainty** and **hallucination**, prevalent in the outputs of GPT-4, since we manually provided the test cases following a prompt-based approach. It commonly produces different results for the same input example. It also generated outputs that were not part of the input text and overlooked the initial prompt regarding the schema (label) information. To ensure an accurate evaluation, we had to manually check if it adhered to the schema and had to intervene to mitigate hallucinations.

The above situation mandates the need for more granular evaluation metrics like *sensitivity* score of Table VIII that takes into account the number of missed predictions (FN) in the ground truth as shown in Table VII. Both of these tables depict fine-grained class-wise results in order to have a deeper understanding of the model performance when it comes to the recognition of individual concepts. To begin with, in Table VII we can see that the most recent LLM based SOTA UniNER model fails to detect even a single entity for the **under-representative class** of '*Composition*'. Even though this UniNER model was trained with multiple clinical and chemical-related datasets [98]–[100]. Another limitation of LLMs is the **limited context window**. The UniNER, in this case, has a context length of a maximum of $2,048$, meaning it is unable to parse any text beyond this token length. While that is the case for LLMs, the standard LM-SD, which was fine-tuned with the structured data, is **mostly biased towards the most likely class**, which is '*Disease*' in this case. Out of LM-SD's $2,421$ total predictions, 819 are labelled as '*Disease*'. This proves the bias of LLM/LM when it comes to detecting more complex concept categories having an imbalance sample space. THOR, however, is more consistent across the classes; thus, the overall **sensitivity score** was the highest for THOR (τ=0.8). The most performant LM-Human, here, comes in third place, scoring 62.24% in terms of sensitivity (Table VIII) with relatively higher number of FN (839). Both LLMs (GPT-4, UniNER) have a high FN compared to our approach.

Compute Requirements. Both THOR and the Baseline model can be run on a regular CPU without bottlenecks. In contrast, the requirements for fine-tuning a standard language model such as LM-SD and LM-Human are comparatively high. We used an Nvidia RTX 3060 GPU with 6GB of GDDR6 GPU memory (VRAM) and 3840 CUDA cores, which lead to an average time of 1–3 hours for fine-tuning. On the other hand, the prompt-based zero-shot UnivNER LLM model needed an A100 GPU with 40GB of VRAM just for inferencing over the test set. It took 34–56 minutes to produce the outputs of the test documents. Fine-tuning was not an option since, according to the authors, it needs 8xA100 GPUs and over a month of training in order to fine-tune it with a custom dataset. For the GPT-4, we used the ChatGPT with the latest GPT-4 model to do the inferencing on our test data. Normally, GPT-4 costs around $\$0.06$ for inputs while $\$0.12$ for outputting 1K tokens. Training these models from scratch was deemed infeasible since it requires approximately 25,000 A100 GPUs over a period of 100 days.

B. Experiment-2: Manual Vs. Automatic Annotation

Since LM-Human utilizes contextually rich annotated text data, in this experiment, we want to scrutinize the minimum amount of annotated text data required to make LMs outperform THOR. Thus, the intention is to know the minimum manual effort needed in terms of time and resources (number of annotators) to annotate a representative dataset, which can produce better results compared to our approach. Indirectly, this experiment challenges the credibility of such SOTA supervised approaches in real-world data integration scenarios. Table IX shows the minimum and maximum time taken for annotating single instances of disease, document, and tokens. It also gives the approximate total duration of hours it took to annotate the whole dataset. Three *annotators* having specialized knowledge annotated the dataset individually. A separate expert with linguistic knowledge re-assured the proper annotation in the case of disagreement. Each annotator spent *3–6* hours per day, for well over 3 months for annotation. Therefore, the **human effort of annotating** such a dataset is quite impractical on a case-by-case basis.

Next, in Table X, we analyzed how the performance would vary compared to THOR if we divided the dataset into smaller

TABLE VII

CONCEPT WISE FINE-GRAINED RESULTS BASED ON PREDICTED ENTITIES (PRED), CORRECT PREDICTIONS (TP), MISSED PREDICTIONS (FN)

Concept	**Ground Truth**	**Baseline**			**UnivNER**			**GPT-4**			**LM-Human**			**LM-SD**			**THOR (τ = 0.8)**		
		Pred	**TP**	**FN**	**Pred**	**TP**	**FN**	**Pred**	**TP**	**FN**	**Pred**	**TP**	**FN**	**Pred**	**TP**	**FN**	**Pred**	**TP**	**FN**
Anatomy	369	203	123	246	317	196	173	324	176	193	217	179	190	432	206	163	386	233	136
Cause	47	23	19	28	42	31	16	85	39	8	42	37	10	133	43	4	106	36	11
Complication	384	122	119	265	210	197	187	216	209	175	212	205	179	244	244	140	345	304	80
Composition	65	12	6	59	**0**	**0**	65	44	17	48	18	17	48	90	17	48	30	14	51
Diagnosis	141	25	25	116	22	11	130	140	68	73	93	89	52	80	80	61	102	84	57
Disease	410	109	98	312	262	224	186	172	150	260	379	367	43	**819**	344	66	347	277	133
Medicine	376	69	65	311	66	60	316	157	141	235	199	191	185	249	235	141	248	220	156
Precaution	72	40	28	44	27	25	47	114	52	20	73	54	18	137	58	14	140	45	27
Riskfactor	136	58	45	91	88	73	63	186	85	51	85	84	52	91	86	50	163	80	56
Surgery	85	11	11	74	29	26	59	49	31	54	63	57	28	39	39	46	80	73	12
Symptom	137	53	49	88	209	108	29	237	121	16	113	103	34	107	104	33	122	98	39
Overall	2222	725	588	**1634**	1272	951	**1271**	1724	1089	**1133**	1494	1383	**839**	2421	1456	**766**	2069	1464	**758**

TABLE VIII

COMPARISON OF MODELS BASED ON SENSITIVITY SCORE

Concept	**Sensitivity**					
	Baseline	**UniNER**	**GPT-4**	**LM-Human**	**LM-SD**	**THOR**
Anatomy	33.33%	53.12%	47.70%	48.51%	55.83%	63.14%
Cause	40.43%	65.96%	82.98%	78.72%	91.49%	76.60%
Complication	30.99%	51.30%	54.43%	53.39%	63.54%	79.17%
Composition	9.23%	**0.00%**	26.15%	26.15%	26.15%	21.54%
Diagnosis	17.73%	7.80%	48.23%	63.12%	56.74%	59.57%
Disease	23.90%	54.63%	36.59%	89.51%	83.90%	67.56%
Medicine	17.29%	15.96%	37.50%	50.80%	62.50%	58.51%
Precaution	38.89%	34.72%	72.22%	75.00%	80.56%	62.50%
Riskfactor	33.09%	53.68%	62.50%	61.76%	63.24%	58.82%
Surgery	12.94%	30.59%	36.47%	67.06%	45.88%	85.88%
Symptom	35.77%	78.83%	88.32%	75.18%	75.91%	71.53%
Overall	**26.46%**	42.80%	49.01%	62.24%	65.53%	**65.89%**

TABLE IX

ANNOTATION EFFORTS IN TERMS OF MINIMUM AND MAXIMUM TIME.

Single Disease	**Single Doc.**	**Single Token**	**Total Duration**
80m – 150m	7m – 25m	8s – 13s	600+ Hours

sizes and fine-tuned the same LM-Human model, but this time with increasing size of the dataset. We divide the dataset based on the number of diseases, starting with a model LM-Human-1 fine-tuned with just the annotated documents of a single disease yielding an F1 score of 0.32. Note that, here, we calculated the value of the column 'Annotation Time(s)' based on the maximum annotation time of 13s for a single token according to Table IX. Thus, to interpret, in order to get an F1 score of 0.32, we needed to train an LM with human-annotated data that took around $12,649$s ($\sim$ 3.5h) per annotator.

Since this single task-specific training information about the disease was not enough for the LM to reach THOR's level of performance, we increased the number of diseases until we found the cutting point where LM-Human improved THOR's best performance ($\tau = 0.7$). This point was at a total of 124 disease-related documents for 20 diseases.

TABLE X

PERFORMANCE VS. ANNOTATION EFFORT ANALYSIS FOR LM-HUMAN FINE-TUNED ON DIFFERENT SIZES OF DISEASE A-Z AGAINST THOR

Model Name	**Fine-Tuning Dataset Size**				**F1**	**Annotation Time (s)**
	Disease	**Docs.**	**Entities**	**Words**		
THOR (τ = 0.7)	–	–	4706	14010	**0.56**	0
LM-Human-1	1	8	97	973	0.32	12649
LM-Human-10	10	65	748	8617	0.47	112,021
LM-Human-15	15	93	1059	11886	0.55	154,518
LM-Human-20	**20**	**124**	**1384**	**15090**	**0.60**	**196,170**
LM-Human-240	240	1438	18539	168816	0.66	2,194,608

This phenomenon is illustrated in Fig. 8, where we can see the effect of annotation time required in order to get the level of performance (left) beyond our approach results (THOR) together with the number of documents needed (right). In conclusion, for LMs, we need approximately 55 hours of manual annotation time per annotator, along with around 4 human annotators, to annotate and oversee the annotation process to reach what THOR achieves without any human intervention. This is without accounting the time required to reach agreement (a must for rigorous annotations). Further, this annotation is, however, based on a specific schema consisting of fixed concept labels. If somewhere along the line, we need to extend the schema with new concept categories (even for a single one), the whole annotation process has to be repeated, and the model needs to be fine-tuned with the new annotations. This is a major setback for LMs concerning data integration since schema evolution is a common phenomenon.

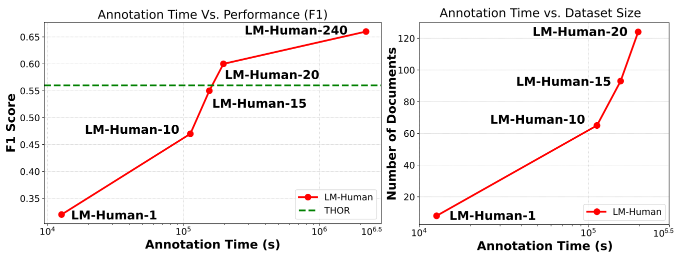


Fig. 8. Annotation Efforts in terms of Time and Performance Gain for LM-Human Models Fine-tuned on Incremental Size of Disease A-Z Dataset

C. Experiment-3: Generalizability

We conducted this experiment to assess the generalizability of THOR and its alternatives. The **Résumé** dataset resembles the typical data an organization may have available (in this case, human resources data containing job-seekers information, which many companies are nowadays analyzing).

The comparative results of Table XI show that even our most strict precision-oriented model THOR (τ=1.0) had the highest **recall** ($R = 0.40$) compared to the alternatives while keeping a steady precision ($P = 0.33$) and F1-score (0.36). Our approach provided the highest number of correct predictions (1244) while the UniversalNER model had the lowest one (188) among all. THOR's and GPT-4 F1-score were very close, but we observe a better correct prediction rate against the ground truth by THOR $(1244/2140)$ compared to GPT-4 $(1030/2140)$. Figure 9 also shows this trend where our approach has the lowest number of false negatives ($FN = 896$). The results also show that the LM-Human model scored low when supplied with the same amount of training data as our previous experiment (test set of 20), showing that the performance of LMs drops drastically when it comes to unseen data, as discussed in Section II.

TABLE XI
THOR (TOP-3 PRECISION) VS. OTHER MODELS: COMPARATIVE OVERALL RESULTS ON RÉSUMÉ DATASET FOR GENERALIZABILITY

Model Name	**Ground Truth Entities**	**Predicted Entities**	**Correct Predictions (TP)**	**Incorrect Predictions (FP)**	**P**	**R**	**F1**
THOR (τ = 0.8)		4416	1606	2819	0.24	**0.50**	0.33
THOR (τ = 0.9)	2140	3296	1420	1876	0.29	0.44	0.35
THOR (τ = 1.0)		2541	**1244**	1297	0.33	0.40	0.36
Baseline		1102	304	798	0.15	0.08	0.10
LM-SD		1045	529	516	0.26	0.12	0.17
GPT-4	2140	2130	1030	1100	0.42	0.38	**0.40**
UniNER		312	**185**	127	0.51	0.07	0.12
LM-Human		506	426	80	0.71	0.17	0.27

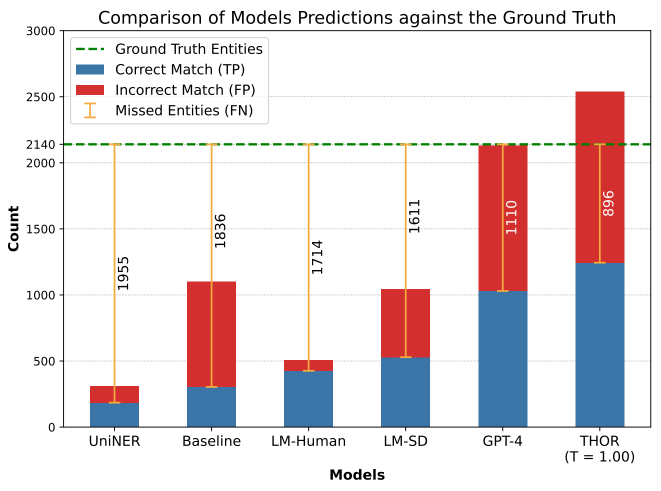


Fig. 9. Bar Chart Representing Overall Prediction Counts Based on TP, FP, and FN against the Ground Truth Entities on Résumé Dataset

We further analyzed the models predictive abilities on a fine-grained level by measuring how good they are across concepts. To that end, Figure 10 shows, via a spider graph, that THOR dominates almost all the alternatives while maintaining a **balanced** F1-score across classes. Although GPT-4 performed quite well for 5 out of 12 classes, it showed extremely bad results for '*Worked As*' and '*Years of Experience*'. Indeed, a closer look reveals that GPT-4 F1-score is based on a good performance on 3 generic entities: names of people, universities and companies. In contrast, THOR outperformed or matched the competing methods in 6 out of 12 classes, particularly excelling at identifying rare concepts where our approach stands out. All in all, THOR shows a better sensitivity score than all the other approaches[11].

[11]More detailed scores (P, R, F1, Sensitivity) can be found in our Github: https://github.com/dtim-upc/THOR/tree/main/Results/Generalizability

The results of the generalizability experiment showcase the limitations of cutting-edge NLP techniques (i.e., LMs and LLMs), which are not prepared to deal with in-company data that does not resemble that of open datasets and benchmarks with which they (e.g., GPT4 or UnivNER) are trained with. THOR, instead, shows a good adaptability even for complex and unseen entities.

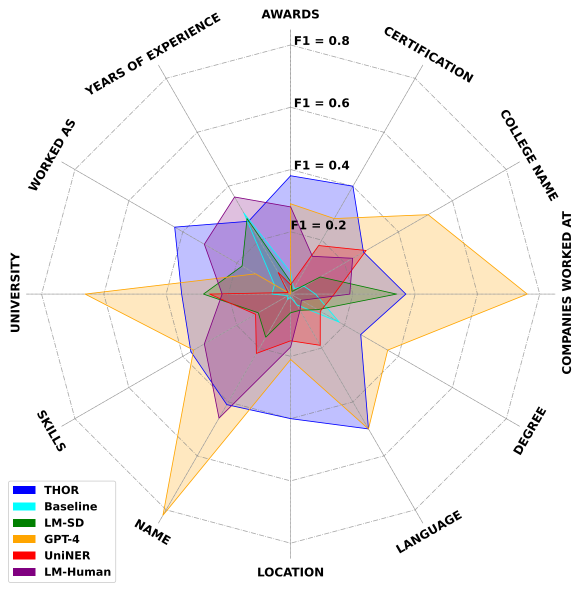


Fig. 10. Fine-grained F1 score per Concept on the Résumé Dataset

VII. Conclusion

We have presented a novel and lightweight approach to address the data sparsity problem for qualitative data, which we redefine as an entity-centric slot-filling problem. THOR's approach discards the need for identification of *subject-verb-object* relations from the text and proposes an entity-centric approach. Framed under the project DEDS (MSCA-ITN GA No 955895) these results prove that this greatly reduces the complexity (both in human effort and computational resources) of structuring information from unstructured text through conceptualization while offering a fresh look at mitigating the problem of data sparsity in integrated data. For our future work, we will explore means to reduce the number of false positives in our approach, specially for high recalls, by further exploring the data integration context and leverage on contextual embeddings (e.g., using knowledge graphs).

Acknowledgments

Md Ataur Rahman is funded by the EC Horizon 2020 research and innovation programme (DEDS: grant agreement No 955895). Sergi Nadal and Oscar Romero are funded by the Spanish Ministerio de Ciencia e Innovación under project PID2020-117191RB-I00/AEI/10.13039/501100011033 (DOGO4ML) and the EC Horizon Europe programme (ExtremeXP: grant agreement No 101093164). The authors would like to acknowledge and thank Md Mohsin Ali, Tonmoy Barua, Srabosthy Das Prama, and Puja Chakraborty for the task of annotating the textual datasets used in the experimental evaluation.

References

[1] M. Lenzerini, "Data integration: A theoretical perspective," in *Proceedings of the Twenty-first ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems, June 3-5, Madison, Wisconsin, USA*, 2002, pp. 233–246. [Online]. Available: https://doi.org/10.1145/543613.543644

[2] M. Lacroix and A. Pirotte, "Generalized joins," *SIGMOD Rec.*, vol. 8, no. 3, pp. 14–15, 1976.

[3] A. Rajaraman and J. D. Ullman, "Integrating information by outerjoins and full disjunctions," in *PODS*. ACM Press, 1996, pp. 238–248.

[4] M. Stonebraker and I. F. Ilyas, "Data integration: The current status and the way forward," *IEEE Data Eng. Bull.*, vol. 41, no. 2, pp. 3–9, 2018.

[5] B. Allison, D. Guthrie, and L. Guthrie, "Another look at the data sparsity problem," in *TSD*, ser. Lecture Notes in Computer Science, vol. 4188. Springer, 2006, pp. 327–334.

[6] A. Y. Xue, J. Qi, X. Xie, R. Zhang, J. Huang, and Y. Li, "Solving the data sparsity problem in destination prediction," *VLDB J.*, vol. 24, no. 2, pp. 219–243, 2015.

[7] I. F. Ilyas and X. Chu, *Data Cleaning*, ser. ACM Books. ACM, 2019, vol. 28.

[8] F. Geerts, G. Mecca, P. Papotti, and D. Santoro, "Cleaning data with llunatic," *VLDB J.*, vol. 29, no. 4, pp. 867–892, 2020.

[9] M. Dallachiesa, A. Ebaid, A. Eldawy, A. K. Elmagarmid, I. F. Ilyas, M. Ouzzani, and N. Tang, "NADEEF: a commodity data cleaning system," in *SIGMOD Conference*. ACM, 2013, pp. 541–552.

[10] E. K. Rezig, M. Ouzzani, W. G. Aref, A. K. Elmagarmid, A. R. Mahmood, and M. Stonebraker, "Horizon: Scalable dependency-driven data cleaning," *Proc. VLDB Endow.*, vol. 14, no. 11, pp. 2546–2554, 2021.

[11] W. Fan, "Data quality: From theory to practice," *SIGMOD Rec.*, vol. 44, no. 3, pp. 7–18, 2015.

[12] T. Rekatsinas, X. Chu, I. F. Ilyas, and C. Ré, "Holoclean: Holistic data repairs with probabilistic inference," *Proc. VLDB Endow.*, vol. 10, no. 11, pp. 1190–1201, 2017.

[13] M. Mahdavi and Z. Abedjan, "Baran: Effective error correction via a unified context representation and transfer learning," *Proc. VLDB Endow.*, vol. 13, no. 11, pp. 1948–1961, 2020.

[14] J. Peng, D. Shen, N. Tang, T. Liu, Y. Kou, T. Nie, H. Cui, and G. Yu, "Self-supervised and interpretable data cleaning with sequence generative adversarial networks," *Proc. VLDB Endow.*, vol. 16, no. 3, pp. 433–446, 2022.

[15] X. Chu, I. F. Ilyas, S. Krishnan, and J. Wang, "Data cleaning: Overview and emerging challenges," in *SIGMOD Conference*. ACM, 2016, pp. 2201–2206.

[16] X. Chu, J. Morcos, I. F. Ilyas, M. Ouzzani, P. Papotti, N. Tang, and Y. Ye, "KATARA: A data cleaning system powered by knowledge bases and crowdsourcing," in *SIGMOD Conference*. ACM, 2015, pp. 1247–1261.

[17] J. a. L. M. Pereira, M. J. Fonseca, A. Lopes, and H. Galhardas, "Cleenex: Support for user involvement during an iterative data cleaning process," *J. Data and Information Quality*, feb 2024. [Online]. Available: https://doi.org/10.1145/3648476

[18] R. Fagin, B. Kimelfeld, F. Reiss, and S. Vansummeren, "A relational framework for information extraction," *ACM SIGMOD Record*, vol. 44, no. 4, pp. 5–16, 2016.

[19] ——, "Document spanners: A formal approach to information extraction," *J. ACM*, vol. 62, no. 2, pp. 12:1–12:51, 2015. [Online]. Available: https://doi.org/10.1145/2699442

[20] S. Sarawagi, "Information extraction," *Found. Trends Databases*, vol. 1, no. 3, pp. 261–377, 2008.

[21] E. Schneider, R. M. Rivera-Zavala, P. Martinez, C. Moro, and E. Paraiso, "UC3M-PUCPR at SemEval-2022 task 11: An ensemble method of transformer-based models for complex named entity recognition," in *Proceedings of the 16th International Workshop on Semantic Evaluation (SemEval-2022)*, G. Emerson, N. Schluter, G. Stanovsky, R. Kumar, A. Palmer, N. Schneider, S. Singh, and S. Ratan, Eds. Seattle, United States: Association for Computational Linguistics, Jul. 2022, pp. 1448–1456. [Online]. Available: https://aclanthology.org/2022.semeval-1.199

[22] M. Keymanesh, A. Benton, and M. Dredze, "What makes data-to-text generation hard for pretrained language models?" in *Proceedings of the 2nd Workshop on Natural Language Generation, Evaluation, and Metrics (GEM)*, A. Bosselut, K. Chandu, K. Dhole, V. Gangal, S. Gehrmann, Y. Jernite, J. Novikova, and L. Perez-Beltrachini, Eds. Abu Dhabi, United Arab Emirates (Hybrid): Association for Computational Linguistics, Dec. 2022, pp. 539–554. [Online]. Available: https://aclanthology.org/2022.gem-1.50

[23] L. Chen, M. Zaharia, and J. Zou, "How is chatgpt's behavior changing over time?" *arXiv preprint arXiv:2307.09009*, 2023.

[24] D. Arora, H. G. Singh *et al.*, "Have llms advanced enough? a challenging problem solving benchmark for large language models," *arXiv preprint arXiv:2305.15074*, 2023.

[25] W. Zhou, S. Zhang, Y. Gu, M. Chen, and H. Poon, "Universalner: Targeted distillation from large language models for open named entity recognition," 2023.

[26] B. Yuan, Y. He, J. Davis, T. Zhang, T. Dao, B. Chen, P. S. Liang, C. Re, and C. Zhang, "Decentralized training of foundation models in heterogeneous environments," *Advances in Neural Information Processing Systems*, vol. 35, pp. 25 464–25 477, 2022.

[27] F. Petroni, A. Piktus, A. Fan, P. Lewis, M. Yazdani, N. De Cao, J. Thorne, Y. Jernite, V. Karpukhin, J. Maillard, V. Plachouras, T. Rocktäschel, and S. Riedel, "KILT: a benchmark for knowledge intensive language tasks," in *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. Online: Association for Computational Linguistics, Jun. 2021, pp. 2523–2544. [Online]. Available: https://aclanthology.org/2021.naacl-main.200

[28] M. Glass, G. Rossiello, and A. Gliozzo, "Zero-shot slot filling with dpr and rag," *arXiv preprint arXiv:2104.08610*, 2021.

[29] P. Jovanovic, S. Nadal, O. Romero, A. Abelló, and B. Bilalli, "Quarry: a user-centered big data integration platform," *Information Systems Frontiers*, vol. 23, no. 1, pp. 9–33, 2021.

[30] S. Yang and J. K. Kim, "Statistical data integration in survey sampling: A review," *Japanese Journal of Statistics and Data Science*, vol. 3, pp. 625–650, 2020.

[31] M. Allen and D. Cervo, "Data quality management," in *Multi-Domain Master Data Management*. Elsevier, 2015, pp. 131–160.

[32] T. D. Pigott, "A review of methods for missing data," *Educational research and evaluation*, vol. 7, no. 4, pp. 353–383, 2001.

[33] S. I. Khan and A. S. M. L. Hoque, "Sice: an improved missing data imputation technique," *Journal of big Data*, vol. 7, no. 1, pp. 1–21, 2020.

[34] S. Jäger, A. Allhorn, and F. Bießmann, "A benchmark for data imputation methods," *Frontiers in big Data*, vol. 4, p. 693674, 2021.

[35] A. Jadhav, D. Pramod, and K. Ramanathan, "Comparison of performance of data imputation methods for numeric dataset," *Applied Artificial Intelligence*, vol. 33, no. 10, pp. 913–933, 2019.

[36] R. Bruni, C. Daraio, and D. Aureli, "Imputation techniques for the reconstruction of missing interconnected data from higher educational institutions," *Knowledge-Based Systems*, vol. 212, p. 106512, 2021.

[37] F. Biessmann, D. Salinas, S. Schelter, P. Schmidt, and D. Lange, "" deep" learning for missing value imputationin tables with non-numerical data," in *Proceedings of the 27th ACM international conference on information and knowledge management*, 2018, pp. 2017–2025.

[38] S. Chen, S. Yang, and J. K. Kim, "Nonparametric mass imputation for data integration," *Journal of survey statistics and methodology*, vol. 10, no. 1, pp. 1–24, 2022.

[39] L. Ren, T. Wang, A. S. Seklouli, H. Zhang, and A. Bouras, "A review on missing values for main challenges and methods," *Information Systems*, p. 102268, 2023.

[40] V. Romero and A. Salmerón, "Multivariate imputation of qualitative missing data using bayesian networks," in *Soft Methodology and Random Information Systems*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2004, pp. 605–612.

[41] J. Fries, S. Wu, A. Ratner, and C. Ré, "Swellshark: A generative model for biomedical named entity recognition without labeled data," *arXiv preprint arXiv:1704.06360*, 2017.

[42] P. Exner and P. Nugues, "Entity extraction: From unstructured text to dbpedia RDF triples," in *Proceedings of the Web of Linked Entities Workshop, Boston, USA, November 11, 2012*, 2012, pp. 58–69. [Online]. Available: https://ceur-ws.org/Vol-906/paper7.pdf

[43] A. P. Quimbaya, A. S. Múnera, R. A. G. Rivera, J. C. D. Rodríguez, O. M. M. Velandia, A. A. G. Peña, and C. Labbé, "Named entity recognition over electronic health records through a combined dictionary-

based approach," *Procedia Computer Science*, vol. 100, pp. 55–61, 2016.

[44] S. Zhao, "Named entity recognition in biomedical texts using an hmm model," in *Proceedings of the international joint workshop on natural language processing in biomedicine and its applications (NLPBA/BioNLP)*, 2004, pp. 87–90.

[45] D. Li, G. Savova, and K. Kipper, "Conditional random fields and support vector machines for disorder named entity recognition in clinical texts," in *Proceedings of the workshop on current trends in biomedical natural language processing*, 2008, pp. 94–95.

[46] T. Rocktäschel, M. Weidlich, and U. Leser, "Chemspot: a hybrid system for chemical named entity recognition," *Bioinformatics*, vol. 28, no. 12, pp. 1633–1640, 2012.

[47] R. Leaman, C.-H. Wei, and Z. Lu, "tmchem: a high performance approach for chemical named entity recognition and normalization," *Journal of cheminformatics*, vol. 7, no. 1, pp. 1–10, 2015.

[48] A. Lenci and M. Sahlgren, *Distributional semantics*. Cambridge University Press, 2023.

[49] J. D. M.-W. C. Kenton and L. K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," in *Proceedings of NAACL-HLT*, 2019, pp. 4171–4186.

[50] Y. Liu, M. Ott, N. Goyal, J. Du, M. Joshi, D. Chen, O. Levy, M. Lewis, L. Zettlemoyer, and V. Stoyanov, "Roberta: A robustly optimized bert pretraining approach," *arXiv preprint arXiv:1907.11692*, 2019.

[51] E. Alsentzer, J. R. Murphy, W. Boag, W.-H. Weng, D. Jin, T. Naumann, and M. McDermott, "Publicly available clinical bert embeddings," *arXiv preprint arXiv:1904.03323*, 2019.

[52] S. S. Bhowmick, E. C. Dragut, and W. Meng, "Globally aware contextual embeddings for named entity recognition in social media streams," in *2023 IEEE 39th International Conference on Data Engineering (ICDE)*. IEEE, 2023, pp. 1544–1557.

[53] X. Wang, Y. Jiang, N. Bach, T. Wang, Z. Huang, F. Huang, and K. Tu, "Automated concatenation of embeddings for structured prediction," in *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, C. Zong, F. Xia, W. Li, and R. Navigli, Eds. Online: Association for Computational Linguistics, Aug. 2021, pp. 2643–2660. [Online]. Available: https://aclanthology.org/2021.acl-long.206

[54] W. Hu, L. Liu, Y. Sun, Y. Wu, Z. Liu, R. Zhang, and T. Peng, "NLIRE: A natural language inference method for relation extraction," *J. Web Semant.*, vol. 72, p. 100686, 2022. [Online]. Available: https://doi.org/10.1016/j.websem.2021.100686

[55] N. Milosevic and W. Thielemann, "Relationship extraction for knowledge graph creation from biomedical literature," 2022. [Online]. Available: https://arxiv.org/abs/2201.01647

[56] X. Wang, Y. Shen, J. Cai, T. Wang, X. Wang, P. Xie, F. Huang, W. Lu, Y. Zhuang, K. Tu *et al.*, "Damo-nlp at semeval-2022 task 11: A knowledge-based system for multilingual named entity recognition," *arXiv preprint arXiv:2203.00545*, 2022.

[57] X. Lin, H. Li, H. Xin, Z. Li, and L. Chen, "Kbpearl: A knowledge base population system supported by joint entity and relation linking," *Proc. VLDB Endow.*, vol. 13, no. 7, p. 1035–1049, mar 2020. [Online]. Available: https://doi.org/10.14778/3384345.3384352

[58] L. S. Vannur, B. Ganesan, L. Nagalapatti, H. Patel, and M. N. Tippeswamy, "Data augmentation for fairness in personal knowledge base population," in *Trends and Applications in Knowledge Discovery and Data Mining*, M. Gupta and G. Ramakrishnan, Eds. Cham: Springer International Publishing, 2021, pp. 143–152.

[59] H. Ji and R. Grishman, "Knowledge base population: Successful approaches and challenges," in *Proceedings of the 49th annual meeting of the association for computational linguistics: Human language technologies*, 2011, pp. 1148–1158.

[60] D. Sui, C. Wang, Y. Chen, K. Liu, J. Zhao, and W. Bi, "Set generation networks for end-to-end knowledge base population," in *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*. Online and Punta Cana, Dominican Republic: Association for Computational Linguistics, Nov. 2021, pp. 9650–9660. [Online]. Available: https://aclanthology.org/2021.emnlp-main.760

[61] D. Xu, J. Zhou, T. Xu, Y. Xia, J. Liu, E. Chen, and D. Dou, "Multimodal biological knowledge graph completion via triple co-attention mechanism," in *2023 IEEE 39th International Conference on Data Engineering (ICDE)*. IEEE, 2023, pp. 3928–3941.

[62] M. Trajanoska, R. Stojanov, and D. Trajanov, "Enhancing knowledge graph construction using large language models," *arXiv preprint arXiv:2305.04676*, 2023.

[63] D. Rao, P. McNamee, and M. Dredze, "Entity linking: Finding extracted entities in a knowledge base," *Multi-source, multilingual information extraction and summarization*, pp. 93–115, 2013.

[64] G. Angeli, M. J. J. Premkumar, and C. D. Manning, "Leveraging linguistic structure for open domain information extraction," in *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, 2015, pp. 344–354.

[65] R. Clancy, I. F. Ilyas, J. Lin, and D. Cheriton, "Knowledge graph construction from unstructured text with applications to fact verification and beyond," in *Proceedings of the Second Workshop on Fact Extraction and VERification (FEVER), Hong Kong, China*, 2019, pp. 3–7.

[66] J. Mao, Y. Yao, S. Heinrich, T. Hinz, C. Weber, S. Wermter, Z. Liu, and M. Sun, "Bootstrapping knowledge graphs from images and text," *Frontiers Neurorobotics*, vol. 13, p. 93, 2019. [Online]. Available: https://doi.org/10.3389/fnbot.2019.00093

[67] A. C. Anadiotis, O. Balalau, C. Conceicao, H. Galhardas, M. Y. Haddad, I. Manolescu, T. Merabti, and J. You, "Graph integration of structured, semistructured and unstructured data for data journalism," *Information Systems*, vol. 104, p. 101846, 2022.

[68] E. Smith, D. Papadopoulos, M. Braschler, and K. Stockinger, "Lillie: Information extraction and database integration using linguistics and learning-based algorithms," *Information Systems*, vol. 105, p. 101938, 2022.

[69] S. Amer-Yahia, G. Koutrika, M. Braschler, D. Calvanese, D. Lanti, H. Lücke-Tieke, A. Mosca, T. Mendes de Farias, D. Papadopoulos, Y. Patil *et al.*, "Inode: building an end-to-end data exploration system in practice," *ACM SIGMOD Record*, vol. 50, no. 4, pp. 23–29, 2022.

[70] I. Melnyk, P. Dognin, and P. Das, "Knowledge graph generation from text," in *Conference on Empirical Methods in Natural Language Processing*, 2022.

[71] G. Rossiello, M. F. M. Chowdhury, N. Mihindukulasooriya, O. Cornec, and A. M. Gliozzo, "Knowgl: Knowledge generation and linking from text," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 37, no. 13, 2023, pp. 16 476–16 478.

[72] Y. Wilks, "Information extraction as a core language technology," in *Information Extraction A Multidisciplinary Approach to an Emerging Information Technology: International Summer School, SCIE-97 Frascati, Italy, July 14–18, 1997*. Springer, 1997, pp. 1–9.

[73] J. Martínez-Rodríguez, A. Hogan, and I. López-Arévalo, "Information extraction meets the semantic web: A survey," *Semantic Web*, vol. 11, no. 2, pp. 255–335, 2020. [Online]. Available: https://doi.org/10.3233/SW-180333

[74] A. Dunn, J. Dagdelen, N. Walker, S. Lee, A. S. Rosen, G. Ceder, K. Persson, and A. Jain, "Structured information extraction from complex scientific text with fine-tuned large language models," *arXiv preprint arXiv:2212.05238*, 2022.

[75] D. Fernàndez-Cañellas, J. Marco Rimmek, J. Espadaler, B. Garolera, A. Barja, M. Codina, M. Sastre, X. Giro-i Nieto, J. C. Riveiro, and E. Bou-Balust, "Enhancing online knowledge graph population with semantic knowledge," in *The Semantic Web–ISWC 2020: 19th International Semantic Web Conference, Athens, Greece, November 2–6, 2020, Proceedings, Part I 19*. Springer, 2020, pp. 183–200.

[76] D. Jurafsky and J. H. Martin, "Speech and language processing. 3rd," 2022.

[77] N. Kertkeidkachorn and R. Ichise, "An automatic knowledge graph creation framework from natural language text," *IEICE Trans. Inf. Syst.*, vol. 101-D, no. 1, pp. 90–98, 2018. [Online]. Available: https://doi.org/10.1587/transinf.2017SWP0006

[78] J. Pujara and S. Singh, "Mining knowledge graphs from text," in *Proceedings of the Eleventh ACM International Conference on Web Search and Data Mining, WSDM 2018, Marina Del Rey, CA, USA, February 5-9, 2018*, 2018, pp. 789–790. [Online]. Available: https://doi.org/10.1145/3159652.3162011

[79] J. Martínez-Rodríguez, I. López-Arévalo, and A. B. Ríos-Alvarado, "Openie-based approach for knowledge graph construction from text," *Expert Syst. Appl.*, vol. 113, pp. 339–355, 2018. [Online]. Available: https://doi.org/10.1016/j.eswa.2018.07.017

[80] R. C. Fernandez, A. J. Elmore, M. J. Franklin, S. Krishnan, and C. Tan, "How large language models will disrupt data management,"

Proceedings of the VLDB Endowment, vol. 16, no. 11, pp. 3302–3309, 2023.

[81] Y. Zhang, V. Zhong, D. Chen, G. Angeli, and C. D. Manning, "Position-aware attention and supervised data improve slot filling," in *Conference on Empirical Methods in Natural Language Processing*, 2017.

[82] M. Glass, G. Rossiello, M. F. M. Chowdhury, and A. Gliozzo, "Robust retrieval augmented generation for zero-shot slot filling," in *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*. Online and Punta Cana, Dominican Republic: Association for Computational Linguistics, Nov. 2021, pp. 1939–1949. [Online]. Available: https://aclanthology.org/2021.emnlp-main.148

[83] W. Zhou and M. Chen, "Learning from noisy labels for entity-centric information extraction," in *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*. Online and Punta Cana, Dominican Republic: Association for Computational Linguistics, Nov. 2021, pp. 5381–5392. [Online]. Available: https://aclanthology.org/2021.emnlp-main.437

[84] E. M. Bender, T. Gebru, A. McMillan-Major, and S. Shmitchell, "On the dangers of stochastic parrots: Can language models be too big?" in *Proceedings of the 2021 ACM conference on fairness, accountability, and transparency*, 2021, pp. 610–623.

[85] X. Wang, Y. Shen, J. Cai, T. Wang, X. Wang, P. Xie, F. Huang, W. Lu, Y. Zhuang, K. Tu, W. Lu, and Y. Jiang, "DAMO-NLP at SemEval-2022 task 11: A knowledge-based system for multilingual named entity recognition," in *Proceedings of the 16th International Workshop on Semantic Evaluation (SemEval-2022)*. Seattle, United States: Association for Computational Linguistics, Jul. 2022, pp. 1457–1468. [Online]. Available: https://aclanthology.org/2022.semeval-1.200

[86] P. Yin, G. Neubig, W.-t. Yih, and S. Riedel, "TaBERT: Pretraining for joint understanding of textual and tabular data," in *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*. Online: Association for Computational Linguistics, Jul. 2020, pp. 8413–8426. [Online]. Available: https://aclanthology.org/2020.acl-main.745

[87] J. Sedlakova, P. Daniore, A. Horn Wintsch, M. Wolf, M. Stanikic, C. Haag, C. Sieber, G. Schneider, K. Staub, D. Alois Ettlin *et al.*, "Challenges and best practices for digital unstructured data enrichment in health research: a systematic narrative review," *PLOS Digital Health*, vol. 2, no. 10, p. e0000347, 2023.

[88] OpenAI, "Gpt-4 technical report," *ArXiv*, vol. abs/2303.08774, 2023. [Online]. Available: https://api.semanticscholar.org/CorpusID:257532815

[89] Y. Chen, Q. Fu, Y. Yuan, Z. Wen, G. Fan, D. Liu, D. Zhang, Z. Li, and Y. Xiao, "Hallucination detection: Robustly discerning reliable answers in large language models," in *Proceedings of the 32nd ACM International Conference on Information and Knowledge Management*, ser. CIKM '23. New York, NY, USA: Association for Computing Machinery, 2023, p. 245–255. [Online]. Available: https://doi.org/10.1145/3583780.3614905

[90] J. R. Taylor, *Lexical Semantics*, ser. Cambridge Handbooks in Language and Linguistics. Cambridge University Press, 2017, p. 246–261.

[91] R. Jackendoff, "Toward an explanatory semantic representation," *Linguistic inquiry*, vol. 7, no. 1, pp. 89–150, 1976.

[92] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient estimation of word representations in vector space," *arXiv preprint arXiv:1301.3781*, 2013.

[93] R. Weischedel, M. Palmer, M. Marcus, E. Hovy, S. Pradhan, L. Ramshaw, N. Xue, A. Taylor, J. Kaufman, M. Franchini *et al.*, "Ontonotes release 5.0 ldc2013t19," *Linguistic Data Consortium, Philadelphia, PA*, vol. 23, p. 170, 2013.

[94] T. Mikolov, E. Grave, P. Bojanowski, C. Puhrsch, and A. Joulin, "Advances in pre-training distributed word representations," in *Proceedings of the International Conference on Language Resources and Evaluation (LREC 2018)*, 2018.

[95] M.-C. De Marneffe, C. D. Manning, J. Nivre, and D. Zeman, "Universal dependencies," *Computational linguistics*, vol. 47, no. 2, pp. 255–308, 2021.

[96] J. W. Ratcliff, D. Metzener *et al.*, "Pattern matching: The gestalt approach," *Dr. Dobb's Journal*, vol. 13, no. 7, p. 46, 1988.

[97] J.-D. Kim, T. Ohta, Y. Tateisi, and J. Tsujii, "Genia corpus—a semantically annotated corpus for bio-textmining," *Bioinformatics*, vol. 19, no. suppl_1, pp. i180–i182, 2003.

[98] L. Luo, P.-T. Lai, C.-H. Wei, C. N. Arighi, and Z. Lu, "Biored: a rich biomedical relation extraction dataset," *Briefings in Bioinformatics*, vol. 23, no. 5, p. bbac282, 2022.

[99] J. Li, Y. Sun, R. J. Johnson, D. Sciaky, C.-H. Wei, R. Leaman, A. P. Davis, C. J. Mattingly, T. C. Wiegers, and Z. Lu, "Biocreative v cdr task corpus: a resource for chemical disease relation extraction," *Database*, vol. 2016, 2016.

[100] R. I. Doğan, R. Leaman, and Z. Lu, "Ncbi disease corpus: a resource for disease name recognition and concept normalization," *Journal of biomedical informatics*, vol. 47, pp. 1–10, 2014.

[101] Ö. Uzuner, B. R. South, S. Shen, and S. L. DuVall, "2010 i2b2/va challenge on concepts, assertions, and relations in clinical text," *Journal of the American Medical Informatics Association*, vol. 18, no. 5, pp. 552–556, 2011.

[102] B. Nye, J. J. Li, R. Patel, Y. Yang, I. J. Marshall, A. Nenkova, and B. C. Wallace, "A corpus with multi-level annotations of patients, interventions and outcomes to support language processing for medical literature," in *Proceedings of the conference. Association for Computational Linguistics. Meeting*, vol. 2018. NIH Public Access, 2018, p. 197.

[103] J. F. Pinzon, S. Krainikovsky, and R. Samarev, "Limitations of neural networks-based ner for resume data extraction," *Procesamiento del Lenguaje Natural*, vol. 65, pp. 53–58, 2020.

[104] I. Segura-Bedmar, P. Martínez, and M. Herrero-Zazo, "SemEval-2013 task 9 : Extraction of drug-drug interactions from biomedical texts (DDIExtraction 2013)," in *Second Joint Conference on Lexical and Computational Semantics (*SEM), Volume 2: Proceedings of the Seventh International Workshop on Semantic Evaluation (SemEval 2013)*, S. Manandhar and D. Yuret, Eds. Atlanta, Georgia, USA: Association for Computational Linguistics, Jun. 2013, pp. 341–350. [Online]. Available: https://aclanthology.org/S13-2056

[105] A. Esuli and F. Sebastiani, "Evaluating information extraction," in *International Conference of the Cross-Language Evaluation Forum for European Languages*. Springer, 2010, pp. 100–111.

[106] A. V. Aho and M. J. Corasick, "Efficient string matching: An aid to bibliographic search," *Commun. ACM*, vol. 18, no. 6, p. 333–340, jun 1975. [Online]. Available: https://doi.org/10.1145/360825.360855

[107] V. Kocaman and D. Talby, "Biomedical named entity recognition at scale," in *Pattern Recognition. ICPR International Workshops and Challenges: Virtual Event, January 10–15, 2021, Proceedings, Part I*. Springer, 2021, pp. 635–646.

[108] Y. Peng, S. Yan, and Z. Lu, "Transfer learning in biomedical natural language processing: an evaluation of bert and elmo on ten benchmarking datasets," *arXiv preprint arXiv:1906.05474*, 2019.

[109] T. Kojima, S. S. Gu, M. Reid, Y. Matsuo, and Y. Iwasawa, "Large language models are zero-shot reasoners," in *Advances in Neural Information Processing Systems*, S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh, Eds., vol. 35. Curran Associates, Inc., 2022, pp. 22 199–22 213. [Online]. Available: https://proceedings.neurips.cc/paper_files/paper/2022/file/8bb0d291acd4acf06ef112099c16f326-Paper-Conference.pdf

[110] C. Qin, A. Zhang, Z. Zhang, J. Chen, M. Yasunaga, and D. Yang, "Is chatgpt a general-purpose natural language processing task solver?" *arXiv preprint arXiv:2302.06476*, 2023.