Journal of Biomedical Informatics 85 (2018) 106–113

Contents lists available at ScienceDirect

Journal of Biomedical Informatics

journal homepage: www.elsevier.com/locate/yjbin

Trie-based rule processing for clinical NLP: A use-case study of *n-trie*, making the ConText algorithm more efficient and scalable

Jianlin Shi*, John F. Hurdle

Department of in Biomedical Informatics, University of Utah, Salt Lake City, UT, USA

ARTICLE INFO

Keywords:
Natural language processing
Medical informatics applications
Algorithms
Data accuracy

ABSTRACT

Objective: To develop and evaluate an efficient Trie structure for large-scale, rule-based clinical natural language processing (NLP), which we call *n-trie*.

Background: Despite the popularity of machine learning techniques in natural language processing, rule-based systems boast important advantages: distinctive transparency, ease of incorporating external knowledge, and less demanding annotation requirements. However, processing efficiency remains a major obstacle for adopting standard rule-base NLP solutions in big data analyses.

Methods: We developed n-trie to specifically address the token-based nature of context detection, an important facet of clinical NLP that is known to slow down NLP pipelines. N-trie, a new rule processing engine using a revised Trie structure, allows fast execution of lexicon-based NLP rules. To determine its applicability and evaluate its performance, we applied the n-trie engine in an implementation (called FastContext) of the ConText algorithm and compared its processing speed and accuracy with JavaConText and GeneralConText, two widely used Java ConText implementations, as well as with a standalone machine learning NegEx implementation, NegScope.

Results: The n-trie engine ran two orders of magnitude faster and was far less sensitive to rule set size than the comparison implementations, and it proved faster than the best machine learning negation detector. Additionally, the engine consistently gained accuracy improvement as the rule set increased (the desired outcome of adding new rules), while the other implementations did not.

Conclusions: The n-trie engine is an efficient, scalable engine to support NLP rule processing and shows the potential for application in other NLP tasks beyond context detection.

1. Introduction[1]

Algorithmic processing efficiency becomes increasingly important as the size of clinical datasets grows, especially in the era of "Big Data" [1]. In the specific domain of clinical natural language processing (NLP), even small increases in processing throughput are important when handling very large note corpora. An extreme example is the U.S. Veterans Administration VINCI national data warehouse, which contains > 2 billion clinical notes. Divita et al. demonstrated the effect of note processing efficiency in VINCI studies [2]. In their work, 6 million records is considered a representative national sample for many applications. Shaving off 100 ms of processing time per note in their benchmark system (401 ms was their nominal per-note-processing time) would save nearly a week of clock time for a corpus of that size.

With the growing interest in Data Science and Big Data, information extraction and retrieval will continue to be a trend in clinical research and practice [3]. Warehoused clinical data growth, spurred in the U.S. by the HITECH Act [4], has escalated the need for faster and more accurate information processing. To improve clinical NLP processing efficiency, we investigated the bottlenecks in typical processing pipelines. Context detection, a rule-based NLP component (defined below), consumed ~70% of processing time in Divita's report.

In this paper, we show that Trie-based rule processing performs well in an important area of clinical NLP. We also present the n-trie (as in "trie for NLP") engine, an optimized Trie rule-processing engine for NLP and demonstrate its efficacy and processing performance with context detection as the use case, specifically the ConText algorithm [5]. Finally, we discuss the engine's potential in other clinical NLP use cases.

* Corresponding author at: Department of in Biomedical Informatics, University of Utah, 421 Wakara Way, Suite 140, Salt Lake City, UT 84108, USA.
E-mail addresses: jianlin.shi@utah.edu (J. Shi), john.hurdle@utah.edu (J.F. Hurdle).

[1] Abbreviations used in this article: "F1-score" is the harmonic mean of precision and recall and is defined in section 3.2.3; "NLP" is "Natural Language Processing; "IE" is "information extraction".

https://doi.org/10.1016/j.jbi.2018.08.002
Received 7 March 2018; Received in revised form 19 July 2018; Accepted 5 August 2018
Available online 06 August 2018
1532-0464/

2. Background

2.1. Rule-based systems are not dead

Traditionally there are three approaches to information extraction (IE) from clinical text: rule-based, machine learning-based, or a hybrid of the two. Although machine learning exhibits promising performance when provided with enough labeled data and a well-defined target structure, its disadvantages in real-world applications are significant: the inner workings of all but a few models are hard to extract (especially true of the currently popular deep neural network models), they are difficult to maintain (e.g., updating or adapting them requires re-training), they are hard to debug (i.e., one can only reduce errors by tuning parameters or by adopting heuristics), and they are hard to enhance with new domain knowledge [6]. In many IE tasks rule-based systems show comparable or superior extraction performance compared to state-of-the-art machine learning solutions [7–10]. Whether one approach or the other has an advantage regarding human labor cost is debatable. Rule-based approaches are generally labor intensive when developing rules. On the other hand, creating a labeled training set and choosing, in a principled and disciplined way, a mathematical model optimized for a specific NLP task requires significant effort and expertise with the machine learning approach.

2.2. Previous research related to optimizing rule processing execution time

Most rule-based systems use regular expressions to define rules or part of rules. In the rule execution experiment conducted by Reiss et al. [11], regular expression execution time accounts for 90% of the overall running time. Chiticariu et al. described SystemT, a rule processing engine that utilizes a set of optimization strategies to speed up rule execution [9]. The most effective strategy involves prioritizing matching rare elements in rules to reduce the matching workload when the rules contain the logical conjunction of multiple elements. However, this strategy only works when prior statistical knowledge about the IE search space is available.

A related optimization that may potentially improve rule processing time uses character trigram indices [12]. After building the character trigram index, regular expressions can be converted to logic-joined trigram search queries. In this way, rule matching speed can be improved by reducing the search space through limiting regular expression execution within trigram-matched query results. In a sense, this approach resembles a very shallow version of the hash tree algorithm we adopted in the work described below. However, trigram indexing requires extra space and substantial preprocessing.

The third related approach is to utilize specifically designed data structures to optimize the string matching process. Since the fundamental goal of rule processing is matching the rules to target text strings, any string matching algorithm potentially could be adopted. For instance, the Trie structure has been studied to host rule dictionaries in hopes of speeding up processing time. A variety of Tries have been demonstrated to be highly effective at string searching, such as the ternary Trie [13], the Patricia Tries [14], the Cache-Efficient Trie [15], and the Hash Array Mapped Trie (HAMT) [16]. HAMP is the ancestor of our n-trie (see Section 4.1), which has a speed-optimized design. It consists of a root hash table with entries of key/value pairs or pointers to sub-hash tables (see Fig. 1). Each sub-hash table has the same structure as the root hash table.

These earlier Trie data structures, based as they are on character strings, frequently prove less flexible and more inconvenient to use for defining clinical NLP rules. Often an NLP rule needs a component that can represent a specific token (e.g., “ruled out” or “denies”). It is difficult to build an *efficient* character-based Trie for large sets of such rules.

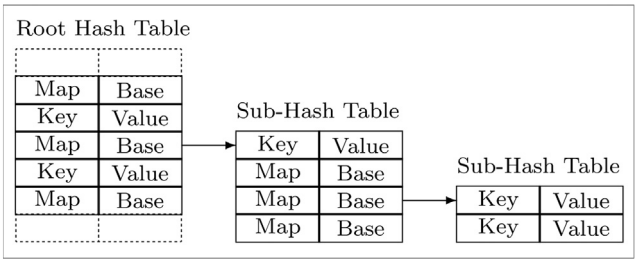

Fig. 1. Hash Array Mapped Trie (HAMT) Structure [16].

2.3. Contextual information

The following subsections briefly review the background of context detection – the use case to demonstrate the n-trie engine. One can think of the term “contextual information” as a set of modifiers associated with a specific mention of a concept. In the domain of clinical information extraction, contextual information typically includes three types of modifiers: *negation* (whether the target concept exists, does not exist, or is possible/uncertain), *experiencer* (whether the target concept refers to the current patient or to someone else), and *temporality* (whether the target concept is currently true, historically true, or hypothetical) [7]. In some studies, the “certainty” (possible/uncertain) is separated as an independent modifier [17,18].

Negation is the best-studied modifier and refers to an assertion about what a patient *does not have*, such as medical conditions or symptoms (e.g., “Patient denies exertional dyspnea”). Previous studies discovered that clinical notes contain a significant number of negative statements [19]. In the corpus used in this study, SemEval-2015 Task 14: Analysis of Clinical Text [20], more than one-fifth of the identified disorder concepts were negated. These negations are often clinically meaningful. They serve critical roles in supporting differential diagnosis generation and treatment planning. For instance, in an ultrasound report, the context of “No atrial septal defect was found” versus the context of “Atrial septal defect is present” would lead to completely different diagnostic and treatment strategies.

2.4. Context detectors

Several rule-based context detectors have been developed and evaluated, such as NegExpander [21], NegFinder [19], and NegEx [22] plus its descendent ConText [5,23]. Rule-based detectors typically define a set of regular expressions that look for trigger terms across a given scope of tokens, usually augmented with additional regular expressions designed to suppress false positives (e.g., double negatives). NegExpander identifies negation words and conjunctions, and asserts the conjunctive noun phrases as negated. However, it cannot adjust the negation scope based on relevant semantic clues, e.g., “although.” NegFinder introduces “negation terminators” to overcome this limitation. Notwithstanding, it cannot handle pseudo-negation words, such as double negations. NegEx and ConText use “pseudo” triggers to deal with these situations. Goryachev et al. [8] compared four different negation detection methods to process discharge summaries, including NegEx, NegExpander, and two other simple machine learning (ML) approaches. The rule-based NegEx (F1-score: 0.89) and NegExpander (F1-score: 0.91) outperformed the two ML approaches (F1-scores: 0.78 and 0.86, respectively). Several ML-based or hybrid systems have been described [17,24–28]. Only NegScope [27], a negation detector evaluated on radiology reports and biomedical abstracts in the Bioscope Corpus [18] and Cogley et al.’s system [28] (a temporality and experiencer detector applied to 120 history and physical notes), outperformed the rule-based systems (Cogley’s system only outperformed ConText in temporality assertions). None of the reports noted above provided any information on processing efficiency.

Although these reported results suggest that the context detection