Computer Languages, Systems & Structures 54 (2018) 1–19

Contents lists available at ScienceDirect

Computer Languages, Systems & Structures

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

Supercombinator set acquired from context-free grammar samples

Michal Sičák*, Ján Kollár

Department of Computers and Informatics, Technical University of Košice, Košice, Letná 904001 Slovakia

ARTICLE INFO

Article history:
Received 8 November 2017
Revised 31 January 2018
Accepted 2 April 2018
Available online 6 April 2018

Keywords:
Supercombinators
Abstract grammars
Context-free grammars

ABSTRACT

We present an algorithm that transforms context-free grammars into a non-redundant set of supercombinators. This set contains interconnected lambda calculus' supercombinators that are enriched by grammar operations. The resulting set is scalable and it can be extended with new supercombinators created from grammars. We describe this algorithm in detail and then we apply it on 62,008 grammar samples in order to find out the properties and limits of acquired supercombinator set. We show that this set has a maximum theoretical limit of possible supercombinators. That limit is the sequence of Catalan numbers. We show that in some cases we are able to reach that limit if we use large enough input data source and we limit the size of supercombinators permitted into the final set. We also describe another benefit of our algorithm, which is the identification of most reoccurring structures in the input set.

1. Introduction

Lambda calculus is a formalism that describes computation with the use of expressions, variables and applications. Combinators are lambda expressions without free variables. We use more restricted form of combinators, supercombinators in our work. The term supercombinator was coined by Hughes in [1] and it means an expression that contains only constants or another supercombinators. In this paper, we show an algorithm that can transform input grammar or a set of grammars into a single supercombinator form that is non-redundant yet retains the descriptive ability of its input grammars.

The main fuel of our work are grammars. We can use them for purposes, which exceed their usual application like the description of a language. The possibilities of wide grammar usage has been presented by Klint et al. in [2]. They argue that grammars are a strong formalism method that are already used in many areas of software engineering. Grammars are also used outside of software engineering field, as Mernik et al. show in [3]. We have presented in our work [4] a way to use grammars as a prime object of internal incremental language evolution.

We have shown in our previous work [5] that any standard Context-free grammar (CFG) can be transformed into a supercombinator form. Which means that we can abstract the structure from the data (represented in grammars as terminal symbols). The experiment performed in the mentioned work showed that we can reduce the amount of grammar elements with this approach rather significantly. We have parsed samples of natural language with the Sequitur [6] algorithm and then converted resulting grammars into a supercombinator form. We have shown that our algorithm abstracts CFGs rather well. In this paper, we are using a source that comes from a more meaningful background, short newspaper articles that

* Corresponding author.
E-mail addresses: michal.sicak@tuke.sk (M. Sičák), jan.kollar@tuke.sk (J. Kollár).

https://doi.org/10.1016/j.cl.2018.04.001
1477-8424/

have already been parsed with the use of Combinatory Categorial Grammar (CCG) [7]. We need a large enough data source that can be converted to a CFG form for further processing. We use samples of CFG grammars that originate from short newspaper articles. Our algorithm depends on the process called grammar inference (see Section 6) with which we can construct grammars from input language samples. We use 10,000 newspaper articles that are a part of Groningen Meaning Bank (GMB) [8] corpus. They already are inferred and are represented as CCG. There are 62,008 sentences in total in those short newspaper articles. Each sentence creates separate grammar. We use 62,008 CFGs extracted from the same number of CCGs as our input data. We explain this process in detail in Section 4.1.

CCGs are often used for meaning related work. However, we do not process these data for some semantic related purpose. Our goal is not to create a new meaning parser, but to analyze the possibilities of a CFG abstraction and to explore their structure. We could even contribute to the field of grammar metrics [9], as the result of our algorithm is a set of supercombinators that represent the structures of input grammars. And those supercombinators are quantifiable and measurable. For more details on this topic, see Section 5. Our ultimate goal is to create a single supercombinator structure that contains data non-redundantly where supercombinators are interconnected. This paper is an extended version of our earlier work [10]. We have extended the algorithm description section and added additional information about the overall transformation process.

Although the algorithm presented in this paper is capable to process virtually any form of a standard CFG, we are leaning towards using it for the natural language processing. Formal grammars describing formal languages tend to be rather short and therefore it might not be so relevant to process them in order to find out the properties of supercombinator set. With the acquisition of a large enough grammar set we can actually see valid results, as we show in Section 4. This of course does not mean that our process is restricted to the natural languages only. In Section 5, we propose how we can use our approach in the area of computer language engineering. The algorithm presented in this paper is entirely written in the functional programming language Haskell.

The main contributions of this paper are:

- We present updated and improved algorithm for supercombinator form acquisition that runs more smoothly than the one from our previous work [5]. The detailed description of its basic functionality is shown in Section 3. We also explain there, why are those changes beneficial for the entire process. The improvements of our algorithm are described in Section 3.7.
- We describe various experiments in Section 4 that we have performed on 62,008 grammar samples taken from 10,000 short newspaper articles included in the GMB corpus. The achieved grammar element reduction performed on input samples obtained from GMB is still significant, as it was in our previous work where we have used Sequitur generated grammars.
- We show in Section 4 that growth of our supercombinator set is limited by the sequence of Catalan numbers [11]. We show theoretical background and statistical analysis in Section 4.4 where we show how this number relates to our results. This limit exists due to the fact that we use supercombinators created from binary CFG rules.
- We show that supercombinators that have been merged to achieve non-redundancy can be tracked during that merge operation in order to acquire more information about the input data. The results in Section 4.5 show that we can identify the most reoccurring structures in the input form, as the structure is directly translated into supercombinators.

2. Motivation

One motivation behind our work is the ability to process grammars, which are the input of our algorithm, into a single supercombinator set. As a grammar needs not to be predefined, it can be acquired from text by a process called grammar inference, our work therefore relies on this process (see Section 6 for more details). We know that we cannot infer a grammar from a set of positive samples purely algorithmically. That has been proven by Gold in [12]. Therefore negative samples,[1] in other words incorrect or ungrammatical sentences, are useful for this process.

But sometimes in human to human communication we do not possess the knowledge about what is and what is not a correct sentence (a negative sample), especially when learning a new language just by listening and repeating it, like little children do. There exist researches that study this phenomena in human to human communication. For example Onnis et al. found out in [13] that people, and especially little children, use cues called variation sets to differentiate utterances as grammatical or not. In the realm of formal languages, by using heuristics like statistical analysis or evolutionary algorithms, we can infer a grammar from positive samples with certain proficiency, see Stevenson and Cordy [14]. And these grammars might have a form of CFG, therefore we can apply our algorithm on them and obtain parallel, non-redundant structure that is scalable. We describe experiments in Section 4 that present the evidence to these claims.

By non-redundancy of a structure we mean that no two elements in this structure are the same (or more precisely equal). It is inherent property of sets, as a set is a collection of distinct objects, i.e. elements that do not repeat themselves. The same applies to our resulting structure, which is a set of supercombinators. But unlike elements in plain sets, our supercombinators have connections between them. We define equality of supercombinators in Section 3.4. In Section 3.5, we

[1] Negative samples are sentences created from the correct alphabet, yet they do not belong to the language we are inferring.