2017

Peirce's Sign Theory as an Open-Source R Package

Authors: Alon Friedman and Erin Feichtinger

Throughout Peirce's writing, we witness his developing vision of a machine that scientists will eventually be able to create. Nadin (2010) raised the question: Why do computer scientists continue to ignore Peirce's sign theory? A review of the literature on Peirce's theory and the semiotics machine reveals that many authors discussed the machine; however, they do not differentiate between a physical computer machine and its software. This paper discusses the problematic issues involved in converting Peirce's theory into a programming language, machine and software application. We demonstrate this challenge by introducing Peirce's sign theory as a software application that runs under an open-source R environment.

Signs vol. 8: pp. 1-24, 2017 ISSN: 1902-8822

Peirce's sign theory as an open-source R package

Alon Friedman[1] and Erin Feichtinger[2]

Abstract:
Throughout Peirce's writing, we witness his developing vision of a machine that scientists will eventually be able to create. Nadin (2010) raised the question: Why do computer scientists continue to ignore Peirce's sign theory? A review of the literature on Peirce's theory and the semiotics machine reveals that many authors discussed the machine; however, they do not differentiate between a physical computer machine and its software. This paper discusses the problematic issues involved in converting Peirce's theory into a programming language, machine and software application. We demonstrate this challenge by introducing Peirce's sign theory as a software application that runs under an open-source R environment.

Keywords: Peirce sign theory, Software, Open source R, package, Fuzzy logic.

1. Introduction

Semiotics, the study of signs, arises from philosophical speculation about signification and language (Chandler, 2004). In the 19th century, deliberation about the meaning of the term "signs" developed in two schools of thought that provided different interpretations. One was promulgated by American philosopher Charles Sanders Peirce, who proposed the study of the sign as "semeiosis." Peirce offered a triadic foundation for the term, in which anything can be a sign as long as someone interprets it as signifying something, referring to, or standing for something other than itself (Peirce, 1931-58, 2.302). Another school of thought, developed by Ferdinand de Saussure (1916/1983, 114), introduces the structuralist point of view which emphasizes the synchronic dimension in the study of language. He proposes a dyadic, or two-part, model of the term sign. The sign, according to Saussure, is made up of the signifier (the mark or sound) and the signified (the concept). Chandler (2004, 65) summarizes the differences between Peirce and Saussure with regard to the term "sign" as follows: the roles of spoken word vs. the written, of the signifier and signified, of time and of language and its meaning. A third perspective was outlined by Petrilli & Ponzio (2002), suggesting that any discussion of semiotics needs to begin with three different categories: a discipline, a theory and a doctrine. Peirce and Saussure, they argue, referred to semiotics as a branch of knowledge,

[1] Assistant professor, School of Information, USF, alonfriedman@usf.edu
[2] M.S. PhD Candidate, Department of Integrative Biology, USF, efeichtinger@mail.usf.edu

and thus as a discipline. Petrilli & Ponzio also denote semiotics as a theory of science, pointing to the discussion of behavioral signs by Morris (1964). To illustrate semiotics as a doctrine, Petrilli & Ponzio cite Sebeok's (1994) non-human signaling and communication systems.

Today's use of computers for communication and discovery in social networking environments raises two important questions for semiotics in the lineage of Pierce: Can we create semiotic machines or software applications based on Peirce's theory? And can this software apply to the domain of social media? We live in a data-driven society preoccupied with communication, where simply using our mobile devices continuously tracks more data about us than we realize. White (1955) called for philosophy to be more concerned with analysis and analytics, and this idea has been very influential in regard to signification and language. In the field of computational semiotics, researchers observe the connection between computation and Peirce's theory. Andersen (1990) calls this discipline computer semiotics, where the aim is to analyze computer systems and their context of use-specific perspective, namely as signs that users interpret to mean something. In comparison, Nöth (2002) calls the discipline the semiotics machine, where a machine is involved in sign processing. According to Nadin (2010) a definition of the field, as to what the semiotics machine needs to or can do, cannot be found in Peirce's own writings. A literature review of Peirce's computational semiotics does not uncover a separation between the physical aspect of semiotics computer machines and software applications.

Currently the majority of social media development focuses on software applications, designed to perform a group of coordinated functions, tasks, or activities for the benefit of the user. These applications aim to address the needs of the growing data-driven society. We do not find any working software application that enacts Peirce's premise of allowing any user to conduct his/her own analysis based on his theories. In order to address this challenge, this paper describes our development of Peirce's theory as an open-source software application package in R. The package aims to provide users with the ability to analyze today's data by applying Peirce's principles of discovery theory.

This article consists of seven sections. In section 2, we discuss the meaning of software applications. In section 3, we outline the R project for statistical computing and its contribution to today's software environments. Section 4 covers Peirce's vision of semiotics and its relationship to machine and software functionality. In section 5, we review the field of computational semiotics literature, focusing on Kumiko Tanaka-Ishii's contribution to the subject, among others. Section 6 demonstrates the capabilities of the Peirce R package. In the last section, section 7, we summarize the results, explain what we have learned and sketch our plans for future developments.

2. Major developments in computing and programming

Computers have touched every part of our lives: the way we work, the way we learn, the way we live, even the way we play. It is almost impossible to go through a single day without encountering a computer in one form or another. As devices, computers are founded on instructions that carry out basic commands that operate the system. The term "computer" was first documented in 1613, originally to describe a human being who performed calculations or computations. The definition of a computer remained relatively unchanged until the end of the 19th century, when the industrial revolution gave rise to machines whose primary purpose was calculating. In 1822, Charles Babbage conceptualized and began developing the Difference Engine, considered to be the first automatic computing machine. The first conceptual computer called the Z1 was created by Konrad Zuse in his parents' living room between 1936 -1938.

Since the invention of Z1, computers have required a means of instructing them to perform a specific task. This means is known as programming language. Computer languages were first composed of a series of steps to wire a particular program; these developed into a series of steps keyed into the computer and then executed. Later, these languages acquired advanced features, such as logical branching and object orientation. In the last 20 years, we have witnessed major changes in mainstream software development and programming languages.

The first major change in programming was the introduction of object-oriented software. Programmers moved away from structured programming to object-oriented programming (OOP) because of the larger CPU and storage resources provided by machine manufacturers. OOP is centered on the concept of "objects," which may contain data, in the form of fields (often known as attributes) and code, in the form of procedures (often known as methods). The unique feature of OOP is that an object's procedures can access and often modify the data fields of the object they are associated with; objects have a notion of "this" or "self." In OOP, computer programs are designed by creating them out of objects that interact with one another. The main component of OOP persists under the object's procedures, which we can access, and we can often modify the data fields of the object with which they are associated. There is significant diversity of OOP languages, but the most popular ones are class-based, meaning that objects are instances of classes, which typically also determine their type.

The second change within the programming domain is the introduction of open-source software, which is often free of charge and allows the user to customize the code. During the 1950s and 1960s, almost all software was produced by academics and corporate researchers working in collaboration, and it was often shared as

public-domain software. During that time, the source code, which can be read by human beings, was generally distributed with the software itself. Users frequently modified the software themselves, since it would run on the same hardware. During the 1980s, computing environments changed, and hardware developers offered computers with different software platforms. The most recognizable application platform is Windows, which the user has to purchase in order to run the software. In 1983, Richard Stallman launched the GNU Project, to write a complete operating system free from constraints on the use of its source code. During that time, Linux, an open-source server developed by Linus Torvaldsm became the most used platform to host websites (personal and public). Wilson (2016), a reporter for *PC Magazine*, ranks Linux as the most reliable source for web hosting. Today, many of the social media companies and sites, such as Firefox and GitHub, are based on the principles of the open-source movement, allowing their users to customize and alter their code.

The third major revolution in the computer industry is cloud computing. Its foundation is Internet-based computing, which shares and provides on-demand computer-processing resources and data for computers and other devices. It is a model for enabling ubiquitous access to a shared pool of configurable computing resources (e.g., computer networks, servers, storage, application, and services), which can be rapidly allotted and released with minimal oversight.

Amid the emergence of "social media" as a new medium reaching global audiences, many researchers still seek to define the term. Many agree that social media refers to web technologies on smartphones and tablet computers that create highly interactive platforms through which individuals, communities and organizations share, co-create, discuss, and modify user-generated content or pre-made content posted online (Correa et. al 2009). Many social media developers rely on open-source applications and cloud computing to provide the social media experience. However, the biggest challenge for researchers and developers is how to handle the data generated every second by social media applications and user input. According to Kaplan and Haenlein (2010), the traditional statistical methodologies do not provide any new insights into the meaning of the data. The new challenge, they argue, is to develop new methodologies to analyze social media data.

These three changes have greatly affected how we compute with today's software applications. One very popular application that has emerged from open-source and cloud computing is open-source R.

3. The R Project for Statistical Computing

The field of statistical computing became popular even before the first calculating machine appeared. In the 1920s, universities and research labs began to acquire the early IBM mechanical punch card tabulators. During the 1960s, growing interest in statistical computing encouraged the development of large mainframe computers. Software applications were created to serve as spreadsheets, which we are accustomed to seeing today. Daniel Bricklin and Bob Frankston created an interactive visible calculator, also known as a spreadsheet. Bricklin named this software program VisiCalc. However, the market during that time was experiencing rapid changes to which VisiCalc could not adapt. In 1985, Bill Gates introduced the Excel spreadsheet software application, which became a foundation of Microsoft Office. With the success of Microsoft Office in capturing the global market, the Microsoft business model promoted the idea that the end user pays for using its product line.

In the mid 1970s, the S Language for Statistical Computing was conceived by John Chambers, Rick Becker, Trevor Hastie, Allan Wilks and others at Bell Labs. However, the development of S Language stopped during 1980s due to lack of funding. In 1995, two professors from the University of Auckland in New Zealand, Ross Ihaka and Robert Gentleman, revised the S language. Ihaka and Gentleman converted it to open-source code in what was initially known as the GNU project, and named the revised language and project "R," based on their first names, Ross and Robert. Together with 17 other people, they established the R-code Foundation. The development of R allowed its community members to address specific problems that traditional spreadsheet software could not. An important reason for R's popularity is the access to packages developed by individuals. A package is essentially a library of prewritten code designed to accomplish a specific task or a collection of tasks. Today, there are more than 6,000 packages available on the Comprehensive R Archive Network, also known as CRAN.

Recent interest in analyzing large sets of data has also created new problems. This type of data usually appears in unstructured form, making it hard to analyze through conventional techniques and applications. Providing a solution, R allows its community to conduct a number of essential procedures for the effective processing and analysis of extensive data.

With all the benefits of open-source R in statistical computer programming, one merit stands out: transparency. The benefits of transparency in statistical computing include fostering trust with customers and community, setting an example of business and scientific practices, and allowing creative problem-solving out in the open. Through R, the user can share any statistical calculation in the code and data with his/her community to receive feedback and additional recommendations. This practice is not possible with any other standardized statistical applications.

R is a form of OOP. Three types of objects constitute the R language. In every computer language, variables provide a means of accessing the data stored in memory. R does not provide direct access to the computer's memory, but rather provides a number of specialized data structures also known as objects. These objects are referred to through symbols or variables. In R, one can have objects based on expressions. An expression contains one or more statements. A statement is a syntactically correct collection of tokens. Expression objects are also special language objects that contain parsed but unevaluated R statements. This syntax will become more clear when we discuss the code and classification we employ to generate Peirce's sign package.

We selected open-source R as our platform to develop Peirce's sign theory. The R platform allows any user to run our code free of charge, using his/her local machine or a cloud computing base that supports R.

4. Peirce's theories and logic

Charles Sanders Peirce (1839–1914), known for his philosophical theory, was also an innovator in many fields. He considered himself a logician first and foremost, and made major contributions to logic, encompassing much of what is now called epistemology and the philosophy of science. Above all, he saw logic as the formal branch of his theory of semiotics. From 1879 till 1884, Peirce taught as a lecturer on logic at Johns Hopkins University. During that time, Peirce had a number of students in logic. One of his most successful students was Allan Marquand (1853-1924), at that time a Fellow in Philosophy and Ethics at Hopkins. His thesis, supervised by Peirce, concerned the logic of Philodemus. During the 1881-82 academic year, Marquand built a mechanical logical machine that Peirce used for the 1886 article *Logical Machine*. In the article, according to Nadin (2010), Peirce described several important aspects of machine functionality, which can help us to better understand the machine-based structure of Peirce's vision.

Throughout his article, *Logical Machine*, Peirce detailed several important aspects of his understanding of machine functionality. The first is that the machine, as he envisions it, evolves the ability to produce statements of logic automatically; he refers here to "Voyage to Laputa" (from Gulliver's Travels). The second aspect is the subject of reasoning that Peirce calls "the nature of the reasoning process" (Peirce, 1887 p. 165). According to him, every machine is a reasoning machine, and every reasoning machine has two inherent impotencies: the destitution of all originality, and of all initiative. That is, it cannot find its own problems, and it can perform only what it was conceived for (Peirce, 1887, pp. 168-169). According to Nadin (2010),

there is no suggestion that semiotics might be of any relevance to understanding the machine described in Peirce's article. However, he points out that it does provide some insight as to how Peirce perceives the process that machines require to generate logical statements about its observation on sciences.

According to Peirce (1903 and 1911), sciences are either sciences of discovery, sciences of review, or practical sciences. Based on this premises, logic is a science of discovery. According to Bellucci and Pietarinen (2014), under the sciences of discovery, Peirce divided science into mathematics, philosophy and idioscopy. Mathematics studies the necessary consequences of purely hypothetical states of things. Philosophy is concerning matters of fact. while Idioscopy embraces more special physical and psychical sciences, and depends upon philosophy. Philosophy in turn divides into phaneroscopy, normative sciences and metaphysics. Phaneroscopy is the investigation of what Peirce calls the phaneron: whatever is present to the mind in any way. The normative sciences (aesthetic, ethics, and logic) introduce dichotomies, in that they are, in general, the investigation of what ought and what ought not to be. Metaphysics gives an account of the universe in both its physical and psychical dimensions. Since every science draws its principles from the ones above it in the classification, logic must draw its principles from mathematics, phaneroscopy, aesthetics and ethics, while metaphysics, and a fortiori psychology, draw their principles from logic (EP 2, pp. 258-262, 1903).

In the context of his theory, Peirce provided a triadic foundation of the term "sign," in which anything can be a sign as long as someone interprets it as signifying, referring to, or standing for something other than itself (Peirce, 1931-1958, 2.302). Peirce maintains that a "sign" is anything that stands for something in somebody's mind. His triangulation consists of Representamen, Interpretant and Object. The physical form of the sign, for Peirce, stands for the Representamen, which is the form, not necessarily material in nature, that the sign takes. The meaning expressed by a sign, as distinct from the physical form in which it is expressed as Object, is that to which the sign refers. Peirce adds an additional element, the Interpretant, which is the sense made of the "sign" (Peirce, 1931-1958, 3.399).

According to Peirce's model, a "sign" is generated on the basis of another sign, and then another signifying process occurs. This unlimited loop continues to generate additional signs. Umberto Eco uses the phrase "unlimited semiosis" (1976, p. 121) to refer to a potential series (as Peirce was well aware) of successive Interpretants, ad infinitum. Peirce's semiotics also refers to all signs, linguistic or non-linguistic, as components of all forms of meaning. Peirce explains that, "All the universe is perfused with signs, if it is not composed exclusively of signs" (1931-1958, p. 489). For Peirce, signs are not just words, and meaning is not necessarily a product of convention or language (1931-1958, pp. 181-183; pp. 251-252). Peirce further

classified the *Object* element into *icons, indexes,* and *symbols*. He defines an icon as similar to its subject; iconic signs carry some quality of the thing they stand for, just as a portrait stands for a person. An index is physically connected with its object, an indication that something exists. The last element is the symbol, which is linked by convention with its object. Symbols are conventional, like most spoken and written words, and are constrained by a more closed than open interpretation process (Peirce, 1931-1958, 2.408). For Peirce, words operate within conventions and are usually taken to be illustrative rather than verbal signs (Juan, 2004).

According to Nöth (1990), under the phaneroscopic nature, Peirce named three universal categories into:

> 1) Firstness—the mode of being of that which is such as it is, positively and without reference to anything in itself;
>
> 2) Secondness—the mode of being of that which is such as it is, with respect to a second but regardless of any third, which can be described as the representation or *Vorstellung*, a representation that is made by the perceiving subject of the object-in-itself; and,
>
> 3) Thirdness—the mode of being of that which is such as it is, in bringing a second and third in relation to each other.

One of the most important characters under these three universal categories of Peirce's logical thought is that logic becomes coextensive with semeiotic, namely the theory of signs. The sign belongs to the first suborder of logic, the speculative grammar, which the foundation of normative sciences that entail aesthetic, ethics, and logic. However, the sign also expresses the relationship between the object and the concept of it as a relationship that is expressed as a "sign. According to Peirce (1998, 2.483):

> A sign or representamen is something that stands to somebody for something in some respect or capacity. It addresses somebody, which creates in the mind of that person an equivalent sign, or perhaps a more developed sign. That sign which it creates I call the interpretant of the first sign. The sign stands for something, its object. It stands for that object, not in all respects, but in reference to a sort of idea. The relation of an object which is in relation to its object on the one hand and to an interpretant on the other in such a way as to bring the interpretant into a relation to the object corresponding to its own relation to the object.

"Firstness" is the mode of being of that which is such as it is, positively and without reference to anything else (CP 8.328). Its existence is either dependent on its being in the mind of some person, or in the form of sense or thought considered by that person. The next category is the "resistance," or an interruption of the initial awareness. This interruption, which recalls us to an awareness of our state of awareness, is a Secondness. According to Hoopes (1991): "secondness is the dyadic mode which tells something about another object." Thirdness is the relationship between Firstness and Secondness; it is a relationship of linguistic signs. Meaning is not inherent, according to Peirce, but something one makes from signs (Peirce, 1983-1917, 2.489). Expression—verbal, written, or otherwise—is the awareness of awareness, which is the Secondness of Firstness, a Thirdness.

Peirce also provided another classification that consists to the meaning of the signs. Under this classification involves the Representamen, Interpretant, and Object to the Firstness, Secondness, and Thirdness as suborder categories. These are:

- Representamen: Qualisign, Sinsign, Legisign
- Interpretant: Argument, Dicent, Rheme
- Object: Icon, Index, Symbol.

The Qualisign identifies a quality, the Sinsign a specific spatio-temporal thing or event, and a Legisign a general idea, a law, a norm. The Argument is, as the name suggests, the reasoning behind the sign, the dicent is a logical proposition that establishes a relationship between constants, and the rheme is the object of firstness or secondness. The icon demonstrates the qualities of a "dynamical object," an index demonstrates the influence of its "dynamical object," and a symbol is a reference to its "dynamical object" that connects the Icon (Firstness) and Index (Secondness).

Another aspect of Peirce's theory concerns math and statistics. Peirce received rigorous mathematical training from his father, which manifests in his philosophy in decidedly mathematical and symbolic veins. The term "statistics," according to Abelson (1995), has its own style of rhetoric and argument that often entails the ability to measure anything that is countable. According to him, the nature of probability is an essential part of statistical paradigms, which allows us to combine logic to measure the chances that our argument will prevail or prove true. According to Hacking (1980) and Miller (1975), Peirce's theory of probability shines some light on his idea of statistics and probability. Miller (1975) cited a letter Peirce sent out to Samuel P. Langley, where Peirce writes:

> We can calculate mathematically, and therefore deductively, precisely how often an induction conforming to certain conditions will give the true probability to a given degree of accuracy, supposing that true probability to be known; and though not precisely, yet within certain limits, how often such an induction will give the probability within certain limits of accuracy, even if the true probability is not given. Still, that leaves us entirely in the dark as to the probability that the ascertained probability, in any particular case, is within any named limits correct. Indeed, it is doubtful whether any meaning can be attached to such a question. All we have to do is to accept the result of induction provisionally, with a degree of confidence depending on the probable accuracy of the proceeding, without troubling our heads about the probability of the inference ratio, and obtain new observations to confirm or modify that ratio (L409, P. 4).

According to Miller (1975), this passage highlights for us some of the insights of Peirce's "plan of operation" for probability and statistics in the everyday world.

5. Computational Semiotics

During the 1990s the idea of the semiotics machine was discussed by numerous researchers, including Andersen (1990), Figge (1991), Nake (1997) and others. Anderson (1990), in particular, describes the machine as a tool that allows users the ability to analyze their own set of data. However, Nöth (2002) raises the question: What is a semiotics machine? Nöth then outlines the idea that a machine is that which interprets and processes the sign. He also discusses the connection between machine functionality and sign interpretation under Peirce's theory. However, Nöth points out that Peirce held a dyadic understanding of machines. In one aspect, Peirce understood that the machine is capable of participating in the processes of semiosis; however, according to Nöth, Peirce also comprehended that human beings control and operate those machines. In 2010, Nadin adds another important question: Why have computer scientists not developed Peirce's theory in a working machine or software application?

Computational semiotics has become an interdisciplinary field that applies semiotics as its framework to conduct and draw on research in logic, mathematics, cognitive sciences and the theory and practice of computation, in which researchers refer directly and indirectly to semiotics theories. Gudwin (2005) summarized the field as an attempt to emulate the semiosis cycle within a digital computer. Many computational semiotics researchers outline Peirce's theory as a form of analysis that is relevant to their own research. However, none of those researchers have

developed their own software application following Peirce's theory as reported by Nadin (2010):

> Even those (like Peter Bogh Andersen, Berit Holmqvist, Lens F. Jensen, Ronald, Stamper, Kecheng Liu) who eventually ventured into information systems, were not able to produce more than some consideration, using semiotics technology – usually not stickily defined – of no real practical relevance. (Nadin, 2010, p. 156),

The first known attempt to simplify programming development under Peirce's theory came from Kumiko Tanaka-Ishii (2010) in her book, *Semiotics Programming*. The aim of her book is to provide semiotics analysis of computer programs along three axes: 1) the models of signs, 2) kinds of signs, and 3) systems of signs. Based on Petrilli & Ponzio's (2002) paradigm of semiotics ranking, Tanaka-Ishii's discussion of semiotics can be defined as a discipline in which the programming code is a branch of knowledge that possesses universal understanding of software applications, which can be applied in any field of investigation or location. According to Tanaka-Ishii (2010), the transformation of Peirce's triadic theory to object-oriented programming (OOP) needs to apply a name, data, and functionalities to lead us to a working code.

Under (OOP), the "objects," which may contain data in the form of fields, are often known as attributes; and code, in the form of procedures, are often known as methods. The main component of OOP persists under the object's procedures, which we can access, and we can often modify the data fields of the object with which they are associated.

Tanaka-Ishii reports on two major problems she faced regarding the code conversions to Peirce's working code. The first issue was the difficulty in implementing the thirdness principle, and the second was obtaining the description for the sign systems.

Looking closer at the problematic issue of thirdness, we can consider this explanation of Peirce:

> If the universe is thus progressing from a state of all but a pure chance to a state of all but complete determination by law, we must suppose that there is an original, elemental, tendency of things to acquire determinate properties, to take habits. This is the Third or mediating element between chance, which brings forth First and original events, and law which produces sequences or Seconds. (Tanaka-Ishii, 2010, p. 54)

Tanaka-Ishii describes her examination of the differences between firstness, secondness and thirdness by explaining that logical arguments and programming code can illustrate the differences between secondness and thirdness functions. Otherwise, the essence of the thirdness stands in as self-reference. To address the problem, she employs Church (1941) and SKI combinator calculus to examine the transformation functionality of the thirdness as a logical argument task. To illustrate her point, we follow her argument in which she used Church's investigation of Lambda calculus (also known as λ-calculus).

The argument consists of the following components: f, x, fix and/or nonrecursively

* The argument expressions of the form: f x, where f is either fix or a non-recursively defined f
* Nonrescursive definitions of the form y = f or y = x and
* The definition of the function "*fix*."

Under Peirce's universal categories in computing the argument will consist of:

Category	**The logic argument**
Firstness	X of the expression f x
Secondness	F of the expression of f x,
Thirdness	Fix

Table # 1 Peirce's universal categories and their logic arguments

According to Tanaka-Ishii (2010) the outcome from the above table is a computer program that essentially consists of, at most, three term relationships. The function *fix* represents genuine thirdness, whereas the other components provide the ability to transform into relations of secondness and firstiness. However, the table indicates that the essence of thirdness is *fix* and a self-reference container, and that computer languages and logical arguments cannot be portrayed by and implemented in the code. According to Tanaka-Ishii and Ishii (2006, pp. 342), "The issue of self-reference is clearly highlighted in programming language, perhaps more in natural language, because random self-reference can be fatal in computers. Therefore, self-reference in programing has been well-formalized with the utmost care. This view of thirdness is such a restricted target might help understand its essence in natural language as well."

The second problem Tanaka-Ishii reports is that once the description is obtained by the use of signs, it then applies to any content that fulfills the description. A description used in this sense involves an abstraction. This way of considering thirdness as an abstraction illustrates why Peirce considered a sign as representative, since the abstraction consists of reflexively reconsidering content in comparison with similar content. However, in computer science and in computer programming languages in particular, it is hard to achieve this formulation of abstraction. According to Brill (1995), the subject of abstraction was one of the main areas of natural language processing in which researchers could not find a universal solution to formulate it. Nadin (2010) reviews Tanaka-Ishii's work and highlights a few of the mistakes Tanaka-Ishii makes. He calls those points her soft spots. According to Nadin, Tanaka-Ishii's work does not discuss the conflict between natural language and computer science. He explains:

> Assuming that Tanaka-Ishii's book lays claim to being the first to attempt some semiotics foundation for programming, we realize that this is a high-order challenge, especially in view of the fact that while natural language is expressive but not precise, programming languages expected to be precise to the extent of limiting ambiguity (which machines cannot handle). But Tanaka-Ishii does not address this aspect. (Nadin 2010, pp.168-172)

Nadin adds an additional assessment of Tanaka-Ishii's book, referring to Kevin McGee (2011), another reviewer of *Semiotics Programming*. According to McGee, the contribution to semiotics research that focuses on communication can be defined as follows: it "tend[s] to be primarily either semioticians analyzing technology or technologists using semiotics concepts . . . to discuss technology" (McGee, 2011 p. 931). This point is not addressed by Tanaka-Ishii. According to Nadin, Tanaka-Ishii makes comparisons between semiotics and semiology without a deep understanding of their respective conditions. Nadin also points out that Saussure is the master of the synchronic view, whereas Peirce advances the dynamic view. To be clear, Peirce has a triadic trichotomic conception. Within this understanding, one cannot write about applying the trichotomy to the computer (Tanaka-Ishii, 2005), just as one cannot take the types of representation (iconic, indexical, symbolic) and turn them into "kinds of signs." The kinds, in Peirce's view, are the ten classes he defined. Of course, operating within those ten classes would be cumbersome, but this is the only way to make sense of Peirce's comprehensive system.

However, despite his criticism, Nadin also points out that Tanaka-Ishii's work is very important to the development of Peirce's and semiotics programming language. He states, "Tanaka-Ishii's competence in formal languages, programming languages, in

particular, qualifies her as a promising researcher of semiotic implications of the age of computation" (Nadin, 2010 p. 163).
Instead, Nadin (1977 & 2010) proposes that instead of naming it semiotics machine, or semiotics programming, he would call it "semiotics engine" to better capture the complexity of computational semiotics. He defines the semiotics engine as processes that are representations and instantiations. He then outlines his mathematical proof of the equivalence between the Peircean sign and fuzzy automata. Figure 1 captures Nadin's formula.

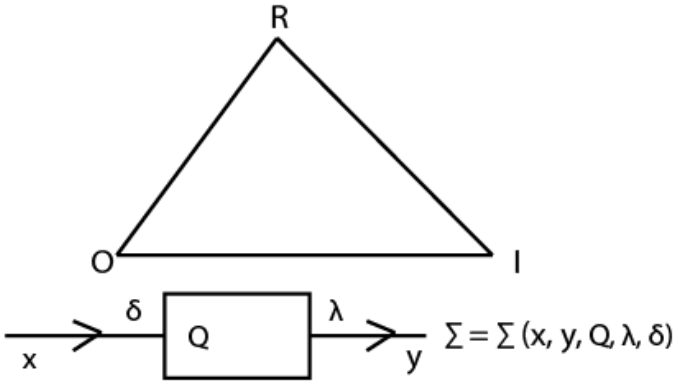

Figure 1. The dynamics of semiotics process by Nadin (1977 and 2010)

The notion of automation, according to Nadin, provides us with the capacity of machine functionality. The fuzzy description of input and output values corresponds to the intention of capturing not only quantitative data, but also qualitative data, often associated with descriptions rather than abstract data. The two transfer functions λ and δ are defined behaviors of non- determination.

The next step under Nadin is the dynamic relation of Peirce's process. In the second step, the sign is the unity of Object, Representamen and Interpretant. The object of representation is further differentiated as immediate and dynamic, as the process of interpretation. Figure 2 captures the dynamics of the semiotics process.

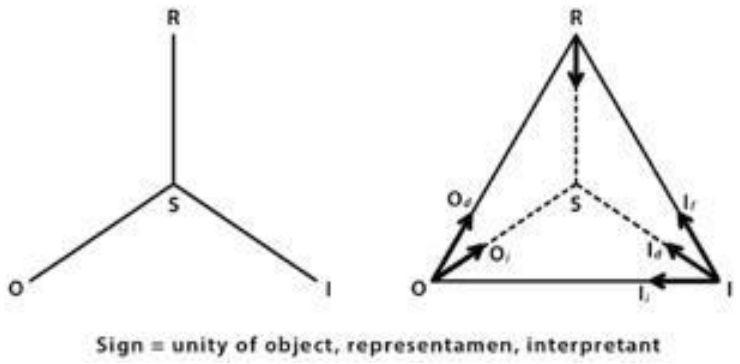

Figure 2. Nadin's dynamic semiotics process diagram.

The following is the last step in this dynamic interpretation of the Object and Interpretant. The diagram below displays the various levels of understanding of the Object represented, which also entails the process of interpretation. Figure 3 displays the dynamic interpretation of the Object and Interpretant, as well as uniquely capturing Peirce's Firstness, Secondness and Thirdness model. This model also provides an alternative discussion by Tanaka-Ishii on the subject.

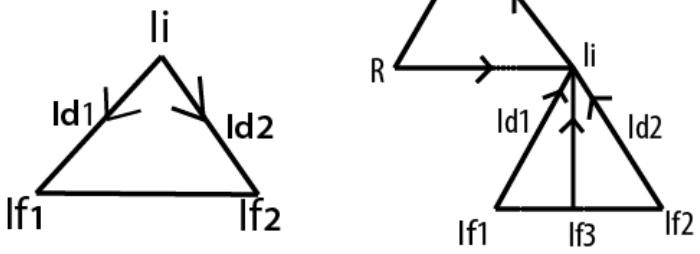

Figure 3. Dynamic interpretation of the Object and Interpretant.

According to Gary et. al (1997), the main advantage of using fuzzy logic in computer programming is to allow to capture the concepts that programmers can identify with, such as complexity, deliberate versus non- deliberate spelling errors, and the degree to which code and comments match. Nadin's utilizing fuzzy logic to capture Peirce's theory allows us to designate input and output values corresponding to Peirce's sign theory and employ it in our open source R package. We employ Nadin's (2010) framework and model to develop our package.

6. Peirce's sign software development in open source R

The package "Peirce's sign" in open source R provides tools to classify and identify relationships between components in data sets by applying Peirce's sign theory of triangulation to qualitative and quantitative data. The main feature of this package is that it implements triangulation from a semiotic perspective, in which the user's goal is to find meaning among components represented with data using Peirce's logic. By "meaning" we are referring to the user's idea of what the signs represent, using the machine as a tool to implement user-defined functions. The machine cannot decide for itself the nature of the relationships among the three components of Peirce's triangulation, e.g. it does not know what words or numbers are, in the way human beings conceive of them. Therefore, the main feature of this package is user-defined input, where the functions in the package can evaluate hypotheses about relationships that are meaningful to human beings.

The three components of the triangulation are Peirce's sign, Object and Interpretant. Currently, triangulation software is available for free via the R package "sigloc" and several websites with triangle calculators. The package "sigloc" employs a function to estimate the locations of radio-tracked animals using GPS coordinates of the transmitter, the receiver and the azimuth bearings, thus applying triangulation of points (Berg, 2015). Triangle calculators on the web return angles and side lengths of triangles given user input of at least three values for angles or sides. Both of these software applications apply triangulation in the general sense, where three points or objects are connected through some relationship implemented through user-defined functions. The program uses one or more parts of these triadic relationships to estimate quantities that the user is interested in. For example, we know rules of mathematics that allow us to compute estimates for the angles and sides of triangles, but users code these functions and the output only has meaning for the user.

The "Peirce's sign" package triangulation is applied in the context of semiosis, or sign theory, in order to address the meaning of the text through the triadic relationship among three repetitive components, the Representamen, the Object and the Interpretant. This three-way relationship among an idea or Representamen, an Object and the Interpretant results in a sign system, the triangle itself. For Peirce, 'the word or sign that man uses *is* the man himself," hence "expression and thought are one," and "every thought is a sign" (Peirce 1958, p. 381) The sign can denote another Representamen; thus, in theory, connections can be made between other triads. The node representing the Interpretant is assumed to be implicit, because it generates the sign through its relationships to the other nodes, the Representamen and the Object. Thus, in the package, the three components are the Representamen, the Object and the sign. The user chooses which variables in a data set correspond

to the Representamen, the Object and the sign based on hypotheses about the relatedness and user-generated meaning from the data components. The four functions in this package apply Peirce's triangulation by looking for associations within each proposed triad that follow the logic of Peirce and the direction of the relationships between the three components. This package makes use of semiosis to find meaning in data and gain insight into causal relationships which is made possible by the underlying logic in Peirce's theory. In the simplest sense, users can examine whether A implies B, which leads to C and vice versa. Furthermore, this functionality in the package allows the user able to examine the interaction between C and B, directly and indirectly. However, the package does not provide a flexible rational agent that perceives its environment and takes actions that maximize its chance of success at some goal. In other words, the four functions in the package do not generate new meaning from the machine's perspective, but simply provide output so the user can determine any potential meaning.

The four functions in "Peirce's sign" package apply triangulation to both numeric and character data. Users input their data in the form of a data.frame in R. Data.frames in R provide the ability to store data in rectangular grids. Each row of these grids corresponds to measurements or values of an instance, while each column is a vector containing data for a specific variable. This means that a data frame's rows do not need to contain, but can hold, the same type of values; they can be numeric, character, logical, or other types of data. Although user data selection must be in the form of a data frame, the methods employed in this package are different based on the type of data. Function 1 is for numerical data, and it makes use of empirical distributions to place each numeric value in a percentage rank based on all values in that column of a dataset, where columns represent each of the three components, and each row represents a triad. This procedure is done for all three columns. Then, the average percentage rank is computed for all three columns at each row, where the rows represent triads. By quantifying where each single data point is within a distribution, we can assess whether numbers in one column are associated with numbers in another. Average percent values for all three components indicate if there is a relationship or not, based on semiosis. Values that depart from 50% in one direction (greater or lesser) indicate an association between the three variables. The user then determines if these relationships are meaningful in the context of his or her hypotheses and goals. Function 2 applies Peirce's sub-classification to the output from function 1, to look for further relationships within the data, as defined by the user's idea of meaning for the data at hand.

The process of applying Peirce's triangulation for character data uses a different procedure than applying triangulation to numeric data. Function 3 first evaluates whether two or more rows are exactly equal, meaning they represent two instances of the same triad. A logical value, true or false, is returned. True indicates that two

rows, which represent triads, have exactly equal values for the object, Representamen and the sign. A return value of false indicates that the rows do not have exact matches at each component. Function 3 also evaluates each column position within triads among rows, corresponding to the Representamen, object and sign, to look for matches at any one of the three components of a triad represented by a row in a data.frame. Another set of logical values is returned indicating whether or not there is an exact match for any of the components within the rows. For example, there could be a triad with cat / fluffy / animal and another with dog / fluffy / animal. The first logical value would be false because the two triads are not equivalent for each of the three components, i.e., the rows are not exactly the same. The next output would give us false / true / true, indicating that the values for the first component are not a match, but the other two components are exact matches between rows/triads. As with the function for numeric data, the user will take this information and determine potential meaning between triads. The output is a summary of how many triads are exact matches, and how many matches there are at each of the three components between triads. From this, the user will get a sense of logical relationships, or lack thereof, between words representing object, Representamen and sign, which facilitates the next step of sub-classification and further identifying more sign processes.

Finally, function 4 of the package generates another sign and signifying process from the original data set after the function 3 has been applied for the initial classification of character data. In other words, the purpose of function 4 is to identify sub-classification or Peirce's firstness, secondness and thirdness. To accomplish this, we make use of R's loop function, which allow the user to iterate a procedure over the entire data set. The code in the loop provides instructions that allow automation of code that is to be repeated. In order to address the sub-classification of Peirce's sign that entails dynamic interpretation of the Object and the Interpretant, we generate a new data.frame from the original dataset. Under this procedure, rows from the first evaluation (function 3) represent instances of triads. Each instance is stored in the first column of the new data.frame represented by a number indicating characteristics held by that triad. The next three columns of the new data.frame are nodes of a triad that represents a sub-classification based on the relationships in the first triads.

The input to the functions for numerical and character data in the package is a data frame with dimensions [n, 3] created from the user's data set with three variables. The numerical function evaluates the rank of integer in a distribution from an empirical distribution function of all values in that column across rows. The average percentage rank for each row is computed then returned for the user to interpret. The output is an object of class "data.frame", same as the input, containing the original rows and columns plus three columns with the percent rank of each entry

and a column with the average percent rank across rows. The expected percentage rank in a numerical distribution of the data is 50%. As with the character function, the interpretation of any meaning as it relates to Peirce's theory is left to the user. Departures from 50% may indicate a pattern. The output object is then used for the sub-classification function.

The first function to process character data looks for exact matches of character strings that consist of each row, representing a potential triad. The output is a list that tells the user where the first duplicated row, if any, is located in the input data frame by indexing the row number [n,], the identity of the columns as character strings with the number of times each row is repeated, and the number of rows in the input value. Unique row values, or non-identical character strings, are not included in this output. The output allows the user to interpret if there are meaningful relationships in the data based on the user's hypothesis about such relationships. The user will then use the sub-classification function. The function uses considers other information associated with the three variables, i.e. other variables in data set, to find relationships within each part of the original triad. The user must decide which new variables to use. The function examines the associated variables and compares the values of these associated variables across rows of data.

To illustrate the overall user functionality in the Peirce's sign package in R, Figure 4 illustrates step by step the overall processes of the package. The first step entails the user must select three variables out of his/her datasets. In step # 2 covers the package analytical analysis functionality we designed in order to run the three variables the user selected in Step 1 under Peirce's paradigm. Step 3 represents the output result the user will receive from Peirce's sign package analysis.

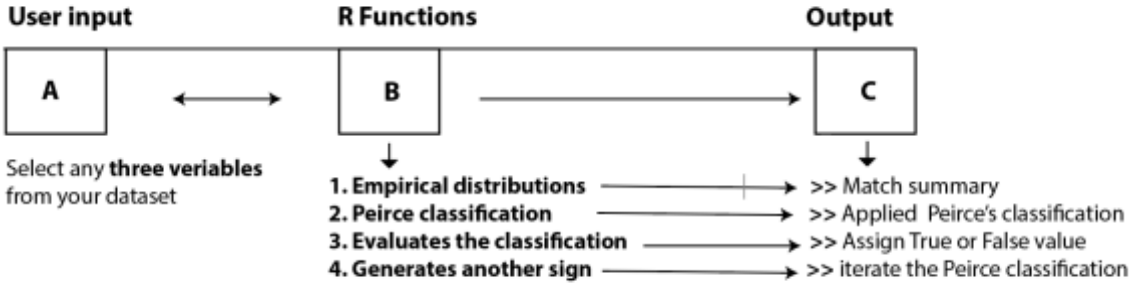

Figure A4. Peirce's sign package functionalities.

In order to demonstrate the functionality of the code in our package, we make use of a data set consisting of tweets during an airing of AMC's "The Walking Dead". The

data consist of the type of interaction between 7,000 users and the Twitter username/handle of each user. For this example, we selected two user tweets, where the data indicated on four types of interactions in this data: mentions, tweets to, replies to, and none. The first user is the object, second user the interpretant and the interaction is the sign. The data, first stored as a CSV file in Excel, were uploaded to R as an object of class "data.frame". We make use of the character function and look for exact matches for each row. In this example, the 199 rows of the input data are evaluated the repeated rows are returned. Below is the output of the function when applied to the Twitter data.

[[1]]
[1] 19
[[2]]

Row ID	Object	Interpretant	Sign	Count
74	countofbluecars	walkingdead_amc	Mentions	2
45	deidbastards	walkingaddicts	Mentions	3
44	deidbastards	walkingdead_amc	Mentions	3
43	deidbastards	walkingdeadarmy	Mentions	3
38	deidbastards	walkingdeadbot	Mentions	2
182	lovatic4everrrr	walkingdead_amc	Mentions	2
19	thatmanjared	walkingdead_amc	Mentions	2
87	tubesdirection	walkingdead_amc	Mentions	2
31	zadf_org	walkingaddicts	Mentions	2

[[3]]
199

The first item, listed under [[1]] is the row number of the first duplicated row in the input data frame. The second item, [[2]], contains the row number, the value of each row (i.e. object, interpretant and sign) and the number of times the row is duplicated. In this example, there are few repeated rows and no more than 3 repetitions for rows. The last item in this output, [[3]], is the total number of rows in the input data set. The sub-classification function requires some more information from the data. For example, if we are interested in learning more about Twitter user patterns, we can add in the user's IP address, the time of the tweet and words contained in the tweet.

The current version of Peirce's sign package, then, uses four functions, where the main function is user-defined. We employ Peirce's idea by allowing the user to select, design and code his/her own triangulation found in the data, based on his or her experience. We selected the open source platform to facilitate this approach. According to Scacchi (2001), the open software movement is heavily involved in its communities. This involvement is required for the development of their software, and for its storage in a persistent, globally accessible form. Our next objective is to

extend these functions to more statistical and probability analysis of Peirce's sub-classification. Furthermore, functions for visualization of the output are under development.

Summary and conclusion

Peirce's theory of signs is a theory of language and reasoning, which holds that all modes of thinking depend on the use of signs. However, Peirce was also known for his contribution to logic and mathematics. In today's dynamic social media environments, statistics and math play an important part as we depend on software applications to communicate and connect. Tanaka-Ishii reported on her attempt to convert Peirce's theory into a software application, encountering two major problems. Nadin (2010) addresses these issues by outlining three formulas, which we employed in order to create the "Peirce's sign" package. We selected open-source R as our platform to develop the package due to its transparency for the R community and for all users. With open-source R, the user is allowed to revise the code and interpret his or her own meaning for data and its classification. Our premise, based on Peirce, was that the user of the package should determine the meaning of the data by selecting the three most important variables. The package asks the user to select the three variables as a form of sign Representamen, Object and Interpretant. The user can add or subtract any additional function in order to fulfill his or her interpretation of the data.

Endnote:

The package was presented at the useR conference in 2016, Stanford University (Friedman, 2016) and is now available for download from GitHub: https://github.com/efeichtinger/Peirce/tree/master/Peirce and soon the package will appear on R CRAN package repository https://cran.r-project.org/web/packages/

References:

Andersen, P. B. (1990). A Theory of Computer Semiotics. New York: Cambridge University Press.

Abelson, R. P. (1995). Statistics as Principled Argument. Hillsdale, N.J.: L. Erlbaum Associates.

Berg, S. (2015). The package "sigloc" for the R software: a tool for triangulating transmitter locations in ground-based telemetry studies of wildlife populations. Bulletin of the Ecological Society of America, (96): 500-507.

Brill, E. (1995). Transformation-based error driven learning and natural language processing: A case study in part-of-speech tagging. Computational Linguistics 21 pp. 543–5.

Chandler, D. (2004). Semiotics for Beginners, Oxford, U.K.: Routledge.

Church, A. (1941). The Calculi of Lambda Conversion. Princeton University Press, Princeton, NJ.

Correa, T., Amber W. H., and Homero G. Z. (2009). Author's Personal Copy Who Interacts on the Web?: The Intersection of Users' Personality and Social Media Use. Computers in Human Behavior 26 (2010) 247–253.

Eco, U. (1976). A Theory of Semiotics. Bloomington: Indiana University Press.

Figge, U. L. (1991). Computersemiotik. Zeitschrifi fr Semiotik, 13(3):321-330.

Friedman, A. (2016) Peirce R sign theory. useR2016 conference, Stanford University, CA.

Gray, A., Sallis, P. and MacDonell, S. (1997). Software Forensics: Extending Authorship Analysis Techniques to Computer Programs. The Information Science Discussion paper series 94/14. University of Otago, Dunedin, New Zealand.

Gudwin, R. R. (2005). Semionics: A Proposal for the Semiotic Modelling of Organizations - In: Kecheng L. (ed.) Virtual, Distributed and Flexible Organizations - Studies in Organizational Semiotics, Kluwer Academic Publishers, 2004, p. 15-34.

Hacking, I. (1980). Grounding Probabilities from Below. In PSA 1980, eds. Asquith, D. A. and R. N. Giere. East Lansing, Mich.: Philosophy of Science Association, 1980, pp. 110-116.

Hoopes, J. (1991). Peirce on Signs. Chapel Hill and London: The University of North Carolina Press.

Juan, E. S. (2004). Charles Sanders Peirce's Theory of Signs, Meaning, and Literary Interpretation. St. John's University Humanities Review vol. 2.2

Kaplan, A. M. and Haenlein, M. (2010). Users of the World, Unite! The Challenges and Opportunities of Social Media. Business Horizons 53 pp. 59-68.

McGee, K. (2011). K. Tanaka-Ishii: Semiotics of Programming (Book review). Artificial

Intelligence, 175 (5-6), 930-931.

Miller, R.W. (1975). Propensity: Popper or Peirce. The British Journal for the Philosophy of Science Vol. 26, No. 2 (Jun., 1975), pp. 123-132.

Morris, C. W. (1964). Signification and Significance: A Study of the Relations of Signs and Values. Cambridge, Mass.: MIT Press. Chap. 1, "Signs and the Act," is reprinted in Charles

Nadin, M. (1977). Sign and Fuzzy Automa. Semiosis 1(5)

Nadin, M. (2010), Information and Semiotics Processes, the Semiotics Computation. Cybernetics and Human Knowing Vol 18(1-2) pp. 153-175.

Nake, F. (1997). Der semiotische Charakter der informatischen Gegenstände. Semiosis 85-90: 24-35.

Nöth, W. (1990). Handbook of Semiotics. Bloomington: Indiana University Press.

Nöth, W. (2002). Semiotics Machines. Cybernetics & Human Knowing 9(1) pp. 5-21

Peirce, C. S. (1931-1958). Collected Papers of C.S. Peirce. Ed. By C. Hartshore, P. Wesis, & A. Burks, 8 vols. Cambridge, MA: Harvard University Press.

Peirce, C. S. (1998). “What is a Sign?” In Houser, N. et al. (Eds.), The essential Peirce: Selected philosophical writings. Vol. 2 (1983-1913), Indiana University Press, Bloomington, pp. 483-91.

Peirce, C. S. (1886 letter). Letter, Peirce to A. Marquand, 1886 December 30, published 1993 in Kloesel, C. et al., eds., Writings of Charles S. Peirce: A Chronological Edition, Vol. 5. Indiana Univ. Press, pp. 421–3.

Peirce, C. S. (1887). Logical Machines. The American Journal of Psychology v. 1, n. 1. Baltimore: N. Murray, pp. 165–70. Google Books Eprint. Reprinted in (1976) The New Elements of Mathematics v. III, pt. 1, pp. 625–32; (1997) Modern Logic 7:71–77, Project Euclid Eprint; and (2000). Writings of Charles S. Peirce v. 6, pp. 65–73.

Petrilli, S. and Ponzio, A. (2002). Vehicles for Semiotics Travels: Two New Handbooks. Semiotica 141(1 /4) pp. 203-350.

Saussure, F.D. (1916/1983). Course in General Linguistic. Trans. Roy Harris, Duckworth, London

Tanaka-Ishii, K and Yuichiro, H. (2006) Thirdness as self-reference in computing. Semiotica 1(4) pp. 327-343.

Tanaka-Ishii, K. (2010). The Semiotics of Programming. New York: Cambridge University Press

White, M.G. (1955). The Age of Analysis: Twentieth Century Philosophers. Houghton Mifflin, London.

Wilson, J.L. (2016). The Best Web Hosting Services of 2016. PC Magazine. From http://www.pcmag.com/article2/0,2817,2424725,00.asp