FEDERAL UNIVERSITY OF CEARÁ

CENTER OF SCIENCE

DEPARTMENT OF COMPUTER SCIENCE

POST-GRADUATION PROGRAM IN COMPUTER SCIENCE

MARCELO BRUNO DE ALMEIDA VERAS

ON THE DESIGN OF SIMILARITY FUNCTIONS FOR BINARY DATA

FORTALEZA

2022

MARCELO BRUNO DE ALMEIDA VERAS

ON THE DESIGN OF SIMILARITY FUNCTIONS FOR BINARY DATA

Thesis submitted to the Post-Graduation Program in Computer Science of the Center of Science of the Federal University of Ceará, as a partial requirement for obtaining the title of Doctor in Computer Science. Concentration Area: Theoretical Computer Science

Advisor: Prof. Dr. João Paulo Pordeus Gomes

FORTALEZA

2022

Dados Internacionais de Catalogação na Publicação
Universidade Federal do Ceará
Biblioteca Universitária
Gerada automaticamente pelo módulo Catalog, mediante os dados fornecidos pelo(a) autor(a)

V584o Veras, Marcelo Bruno de Almeida.
On the Design of Similarity Functions for Binary / Marcelo Bruno de Almeida Veras. – 2022.
62 f. : il. color.

Tese (doutorado) – Universidade Federal do Ceará, Centro de Ciências, Programa de Pós-Graduação em Ciência da Computação , Fortaleza, 2022.
Orientação: Prof. Dr. João Paulo Pordeus Gomes.

1. Similarity measure. 2. Genetic programming. 3. Protein function annotation. 4. Sparse data. I. Título.

CDD 005

MARCELO BRUNO DE ALMEIDA VERAS

ON THE DESIGN OF SIMILARITY FUNCTIONS FOR BINARY DATA

Thesis submitted to the Post-Graduation Program in Computer Science of the Center of Science of the Federal University of Ceará, as a partial requirement for obtaining the title of Doctor in Computer Science. Concentration Area: Theoretical Computer Science

Approved on:

EXAMINATION BOARD

Prof. Dr. João Paulo Pordeus Gomes (Advisor)
Federal University of Ceará (UFC)

Prof. Dr. José Antônio Fernandes de Macêdo
Federal University of Ceará (UFC)

Prof. Dr. José Maria da Silva Monteiro Filho
Federal University of Ceará (UFC)

Prof. Dr. Sabeur Aridhi
University of Lorraine (UL)

Prof. Dr. Engelbert Mephu Nguifo
University Clermont Auvergne (UCA)

This dissertation is dedicated to my two parents, William and Edineusa, who have always fought for my studies, my happiness, and my health.

ACKNOWLEDGEMENTS

This study was financed in part by the Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brasil (CAPES) - Finance Code 001

First and foremost I am extremely grateful to my supervisor Professor Dr. João Paulo Pordeus Gomes for his guidance, continuous support, and patience during my Ph.D. study, which enabled me to complete my research. Also, I would like to thank Lorena, my love and life partner, for her unconditional support and resiliency in adverse circumstances. Additionally, I thank Professor Dr. Heráclito Jaguaribe and Sérgio Girão for their advice, which helped me develop my academic and professional skills. I also thank the MDCC students, my colleagues, and friends Weslley Lioba, Diego Farias, Renan Vieira, and Alisson Alencar for the many discussions and lessons learned and the entire MDCC team for their goodwill and always being available for any questions and requests. Finally, I would like to express my gratitude to my parents, and my brother William for being always supportive.

“Life is precious. Infinitely so. Perhaps it takes a machine intelligence to appreciate that.”

(Alastair Reynolds)

ABSTRACT

The binary feature vector is a widely used representation in many areas of knowledge. They serve to indicate the presence or absence of certain characteristics, therefore, functions that make use of these representations, such as similarity functions, are important to recognize how objects are similar to each other and perform tasks, such as classification, clustering and detection of outliers. The similarity function is a measure that quantifies this similarity and directly influences the performance of a proposed solution. Due to its importance, it is fundamental to properly solve a problem that a good similarity function is used. For choosing a similarity function, two approaches are commonly used: one is to search and analyze existing functions that fit the problem better, and the other is to create a new function with a specialist. In this work, both approaches are examined, and a new proposal is made for each approach outlining both the advantages and disadvantages. In the first one we present a methodology to designing similarity functions and a new function to deal of sparse data, as well as evaluating of proposed function through a series of experiments. In the second one, we propose an automated framework that learns from data to generate similarity function that are appropriate to a given task. This framework was developed to generate functions with theoretical properties necessary for a similarity function. Again, a series of experiments are conducted to asses its importance. We evaluated both studies performances in relation to 63 other similarity functions. Based on the results, we can state that in both cases our proposals were able to outperform classical functions in most of the tested cases.

Keywords: similarity measure; similarity function; genetic programming; protein function annotation; protein structure; sparse data.

RESUMO

O vetor binário de características é uma representação amplamente utilizada em diversas áreas do conhecimento. Elas servem para indicar a presença ou ausência de determinadas características, portanto, funções que fazem uso dessas representações, como funções de similaridade, são importantes para reconhecer como os objetos são semelhantes entre si e realizar tarefas, como classificação, agrupamento e detecção de valores atípicos. A função de similaridade é uma medida que quantifica essa similaridade e influencia diretamente no desempenho de uma solução proposta. Devido à sua importância, é fundamental para resolver adequadamente um problema que uma boa função de similaridade seja utilizada. Para a escolha de uma função de similaridade, duas abordagens são comumente utilizadas: uma é buscar e analisar funções existentes que melhor se ajustem ao problema, e a outra é criar uma nova função com um especialista. Neste trabalho, ambas as abordagens são examinadas, e uma nova proposta é feita para cada abordagem delineando as vantagens e desvantagens. Na primeira apresentamos uma metodologia para projetar funções de similaridade e uma nova função para lidar com dados esparsos, bem como avaliar a função proposta através de uma série de experimentos. Na segunda, propomos um framework automatizado que aprende com os dados para gerar funções de similaridade apropriadas para uma determinada tarefa. Este framework foi desenvolvido para gerar funções com propriedades teóricas necessárias para uma função de similaridade. Novamente, uma série de experimentos são realizados para avaliar sua importância. Avaliamos o desempenho de ambos os estudos em relação a 63 outras funções de similaridade. Com base nos resultados, podemos afirmar que em ambos os casos nossas propostas foram capazes de superar as funções clássicas na maioria dos casos testados.

Palavras-chave: medida de similaridade; função de similaridade; programação genética; anotação da função protéica; estrutura protéica; dados esparsos.

LIST OF FIGURES

LIST OF TABLES

SOURCE CODE LIST

LIST OF ABBREVIATIONS AND ACRONYMS

BSM	Binary Similarity Measures
EBI	European Institute of Bioinformatics
EC	Enzime Comission
GA	Genetic Algorithm
GO	Gene Ontology
GP	Genetic Programming
NLP	Natural Language Processing
ROC	Receiver Operating Characteristic
SSB	Sparse Binary Vectors

LIST OF SYMBOLS

S	Similarity Measure
$\mathcal{X}$	Data Domain
$\bar{X}, X$	Input matrix
Y	Target Vector
x	Binary Vector
x_i	Element i of vector **x**
i, j, k	Indexing value
P	Precision array
R	Recall array
a	Value associated to a pair of binary vectors with number of features that are present in both subjects
b	Value associated to a pair of binary vectors with number of features that are present in the first but not in the second
c	Value associated to a pair of binary vectors with number of features that are present in the second but not in the first
d	Value associated to a pair of binary vectors with number of features that are absent in both
n	Sum of a, b, c and d
$\mathbb{S}$	SSB function
$\mathbb{S}'$	Not normalized version of SSB
δ	Hyperparameter that multiplies a in the exponential part of $\mathbb{S}'$
λ	Hyperparameter that multiplies the inner function $\mathbb{S}'$ of SSB
u, v	Node of a graph
(u, v)	Edge that links u and v
$W_(u, v)$	Weight of the edge (u, v)
$N(u)$	Set of neighbors of u
$L(u)$	Set of labels (annotations) of u

D	Set of domains
d	Protein domain
f	Function which is product of GP process
f_s	Symmetric function that uses f
f_{nn}	Non-negative function that uses f_s
$\mathcal{F}$	Similarity function which is a product of the framework
$sign$	Signal function

CONTENTS

1 INTRODUCTION

Similarity functions can be defined as real-valued functions that quantify the similarity between two objects. Such functions work as building blocks of several pattern recognition methods with application in many domains like bioinformatics (WIJAYA *et al.*, 2016), computer vision (OLIVEIRA *et al.*, 2018) and Natural Language Processing (NLP) (SMARANDACHE *et al.*, 2019). Among all similarities, a family of particular interest is Binary Similarity Measures (BSM). These measures are designed to quantify the similarity between vectors of binary features. Such codification is widely used to represent concepts like presence/absence, yes/no, or true/false (WIJAYA *et al.*, 2016).

As an important part of many algorithms, the choice of a BSM directly influences the performance of a proposed solution. So, in recent years, many researchers concentrated efforts on finding the best measure for a given application. The proposed approaches can be divided into two major directions: finding the best BSM in a set of similarities and designing a new BSM that is tailored to the application.

Comparisons between BSM have been the object of study in many works like in Brusco *et al.* (2021) and Abreu *et al.* (2006). As expected, it is observed that no measure is able to outperform all others in all applications (Wolpert; Macready, 1997). Thus, works like Wijaya *et al.* (2016) and Rácz *et al.* (2018) proposed strategies to automatically select a BSM among a set of measures for a specific application. The authors use hierarchical clustering and Receiver Operating Characteristic (ROC) analysis to select the best measure. The proposal is evaluated in a herbal medicine application. It is important to point out that the success of such approaches is directly related to the set of BSM available.

The second major research direction consists of designing new BSMs by using the knowledge of domain experts. Successful applications of such an approach can be found for domains like software clustering (NASEEM; DERIS, 2017) and interval-valued data (KABIR *et al.*, 2017), among others. In such problems, the proposed BSM was able to outperform all previously proposed general-purpose BSMs.

Although domain-specific BSMs had remarkable results, such strategy depends on the expert knowledge and that may be unavailable in many cases. An alternative solution can be designed by using data to drive the project of a new BSM. Such data-driven BSM construction may provide good results since many nonlinear

mapping methods are available and had been used in various applications. However, an important drawback of a fully automated data-driven BSM design is the absence of guarantees that the proposed BSM has some desirable theoretical properties (Symmetry, non-negativity, and maximality).

The present study explores both approaches mentioned above, it introduces a new BSM for the protein function annotation problem and a data-driven framework to automatically design new BSMs. The new BSM is called Similarity metric for Sparse Binary vectors (SSB) and it was designed to combine important characteristics of the best similarities for the protein function prediction task and take into account the knowledge of specialists in the problem. In the second approach, we propose a framework that builds similarity functions from data. To maintain the properties of symmetry, non-negativity, and maximality within that framework, a function-approximation method and a modular construction method are used.

In both proposals, we performed a series of experiments to evaluate their performances. The experiments were conducted in one or more real-world application scenarios and comparisons were performed with other BSMs available in the literature and widely used in other applications.

1.1 Objectives

The general purpose of this research is to develop a new similarity measure for protein function annotation problems and for problems with available data. This will lead to new functions that will be more effective than existing ones.

Specifically, this study aims to analyze the construction of other BSMs and observe their performance on the same protein problem, aiming to develop a specialized BSM that performs better. This brand-new function needs to respect the three theoretical properties of similarity functions. Furthermore, this work also aims to create a framework for creating new BSMs automatically. With the help of a function-approximation method, the framework will learn from the data and create a new similarity function, and that function will follow the similarity function's properties as well.

1.2 Publications

The following publications were made in the development of this work:

Marcelo B.A. Veras, Bishnu Sarker, Sabeur Aridhi, João P.P. Gomes, José A.F. Macêdo, Engelbert Mephu Nguifo, Marie-Dominique Devignes, Malika Smaïl-Tabbone, **On the design of a similarity function for sparse binary data with application on protein function annotation**, Knowledge-Based Systems, Volume 238, 2022, 107863, ISSN 0950-7051, https://doi.org/10.1016/j.knosys.2021.107863.

In parallel, the following work is about to be submitted: Marcelo B.A. Veras, Sabeur Aridhi, João P.P. Gomes, José A.F. Macêdo, Engelbert Mephu Nguifo, Marie-Dominique Devignes, Malika Smaïl-Tabbone, **Creating Binary Similarity Functions with an Automated Framework**, 2022

1.3 Dissertation structure

The remainder of this dissertation is structured as follows:

- **Chapter 2** provides a detailed introduction to binary similarity, protein function annotation, and genetic programming. As a result of reading this chapter, the reader will be familiar with the three topics and their nomenclatures and will be able to follow the rest of the dissertation.
- **Chapter 3** proposes a new BSM and explains the methodology behind its development, along with an evaluation of the proposed function in comparison with 63 others in the literature.
- The second proposal, described in **chapter 4**, is the creation of an automatic framework to learn from data and construct new similarity functions. The 63 similarity functions are also used for comparing the results of the automatic framework.
- **Chapter 5** Restate the proposal's achievements and summarizes the results of both studies carried out in this dissertation and suggest the next steps to be taken.

2 THEORETICAL BACKGROUND

This chapter presents the theoretical background for the dissertation. Three subjects are most relevant in this research: Similarity functions, Protein Function Annotation, and Genetic Programming (GP). Similarity functions measure the resemblance between two objects, in section 2.1 shows how these functions turn into a metric, their properties, and how it has been used in the literature. In section 2.2 we explain the Protein Function Annotation, its relevance, and how different approaches use the amino acid sequence to predict protein functions, additionally, we provide a description Similarity Measures that can be used to aid in this task. Section 2.3 explains an important tool called GP, which is an evolutionary technique that provides solutions for problems based on biological principles such as the survival of the fittest.

2.1 Similarity Functions

Similarity functions quantify in a real number how similar two objects are. Such measures play a key role in algorithms for pattern analysis applications such as computer vision (WOLF *et al.*, 2009), clustering (CONSONNI; TODESCHINI, 2012a) and outlier detection (NAGARAJA *et al.*, 2020).

Formally as stated by (LESOT *et al.*, 2009), the similarity measure S is a function $\mathcal{X} \times \mathcal{X} \to \mathbb{R}$ where $\mathcal{X}$ is the data space and S verifies the following properties:

- $\forall x, y \in \mathcal{X}, S(x, y) \geq 0$ (Non-negativity)
- $\forall x, y \in \mathcal{X}, S(x, y) = S(y, x)$ (Symmetry)
- $\forall x, y \in \mathcal{X}$ and $x \neq y, S(x, x) \geq S(x, y)$ (Maximality)

Non-negativity means that comparing two objects will not result in a negative value for a similarity. The Symmetry property says that similarities are commutative for any possible pair of elements, that is, changing the input order of the function the result should remain unchanged. Maximality implies that the comparison of two same objects produces a value equal to or greater than the comparison of different objects.

Some authors impose a normalization constraint, requiring the measure to take values between 0 and 1. The normalized version of the general similarity could require the metric to undergo a normalization transformation. We consider the

normalized version of the similarities functions in the rest of this study. However, there are situations where these properties can be loosened to a more general definition, as shown in Hubálek (2008) and Choi *et al.* (2010). In such cases, the similarities do not have the symmetry property, causing an object to be more similar to another depending on the direction of the comparison.

2.1.1 Binary Similarity Measures

A simple way of assessing the similarity or difference between two individuals is to compare their properties. In many situations such properties can be represented in a binary way, denoting the presence/absence of a given property. In such cases, BSM may be used to quantify the similarity among individuals.

Mathematically defining, each subject to be compared is represented as a binary vector **x** $= \{x_1, x_2, x_3, \cdots, x_m\}$, where m is the number of properties, x_k getting the value of 1 if the property k is present and 0 otherwise (for $1 \leq k \leq m$). From this representation, four dependent quantities could be derived from each pair of vectors to aggregate the information about the similarity between them. Let x and y two objects to be compared, **x** and **y** their binary representation respectively and x_i and y_i the information about presence/absence of the property of index i in x and y respectively, we can define the four quantities as:

$a =$ The number of features that both subjects share in common.
$b =$ The number of features that are present in x but absent in y.
$c =$ The number of features that are present in y but absent in x.
$d =$ The number of features that are absent in both subjects.

Moreover, the dependent quantities divide the BSMs into two groups those which do not make use of characteristics that are absent in both subjects (those that use only a, b, and c), and the group of functions which do not (i.e., those that consider a, b, c, and d).

Examples of similarities from the first group:

$$S_J(a,b,c) = \frac{a}{a+b+c} \quad \text{(Jaccard)}$$
$$S_D(a,b,c) = \frac{2a}{2a+b+c} \quad \text{(Dice 1/Czekanowski)}$$
$$S_S(a,b,c) = \frac{a}{a+2b+2c} \quad \text{(SokalSneath 1)}$$
$$S_P(a,b,c) = \frac{ab+bc}{ab+2bc+cd} \quad \text{(Peirce)}$$

Examples of similarities from the second group:

$$S_P(a,b,c,d) = \frac{ad-bc}{\sqrt{(a+b)(a+c)(d+b)(d+c)}} \quad \text{(Pearson Heron 1)}$$
$$S_{Yq}(a,b,c,d) = \frac{ad-bc}{ad+bc} \quad \text{(Yuleq)}$$
$$S_{Yw}(a,b,c,d) = \frac{\sqrt{ad}-\sqrt{bc}}{\sqrt{ad}+\sqrt{bc}} \quad \text{(Yulew)}$$
$$S_T(a,b,c,d) = \frac{na-(a+b)(a+c)}{na+(a+b)(a+c)} \quad \text{(Tarwid)}$$

where $n = a+b+c+d$. Regardless of the group, it is common to see a and d as increasing factors and b and c as decreasing factors in the functions. Modeling the fact that those functions measure the characteristics that the objects share or are absent at the same time.

The similarity measures are motivated by the fact that a pair with a strong similarity score has a positive entailment relationship, and use this premise in methods to build more complex relations and propose solutions. However, according to the famous "no free lunch principle", no similarity function outperforms all others for all applications (Wolpert; Macready, 1997). So, since Jaccard proposed a similarity measure in 1901, many BSMs have been proposed in various fields (CHOI *et al.*, 2010). The great number of available measures is underlined in works like (BELOHLÁVEK *et al.*, 2014), (WIJAYA *et al.*, 2016), (TODESCHINI *et al.*, 2012) and (CHOI *et al.*, 2010). Each of those papers evaluates the performance of more than 50 similarities in different tasks.

2.2 Protein Function Annotation

The identification of protein functions is fundamental in many medical problems including drug design and human disease understanding (BAKHEET; DOIG,

2009a). However, protein function annotation is considered a challenging task since manual annotation is labor-intensive and the number of proteins is increasing exponentially. We notice that the number of entries in UniProtKB, the largest resource for protein sequence and annotation data, evolves rapidly over time. For example, the number of entries in UniProtKB has grown from around 50 million in 2014 to more than 195 million entries in 2020 (SARKER *et al.*, 2020).

In UniProtKB, proteins are described by their amino acid sequences and other information like InterPro domains that are defined as conserved sequence blocks in protein structures (SARKER *et al.*, 2020). One important task supported by the UniProt team at the European Institute of Bioinformatics (EBI) is predicting Enzime Comission (EC) numbers based on protein sequences or protein structures. The EC number is a classification for enzymes that constitute an important class of proteins. This classification is mainly based on the chemical reactions catalyzed by proteins. Recent EC prediction methods include machine learning approaches (LI *et al.*, 2017), sequence encoding (SHEN; CHOU, 2007), domain similarity (SARKER *et al.*, 2020) and structural similarity (YANG *et al.*, 2014).

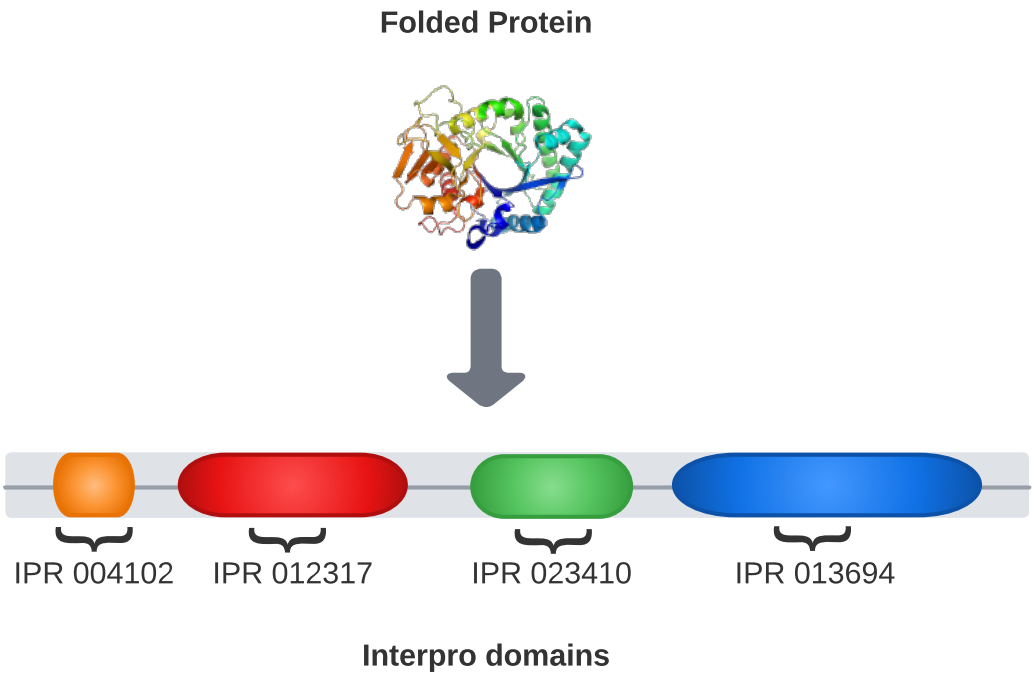
It is widely accepted that protein domains are commonly related to distinct three-dimensional structures and that it often results in a close relationship between protein structure and functions (WIDLAK, 2013; ALBORZI *et al.*, 2017). Hence, many works like (SARKER *et al.*, 2020; ALBORZI *et al.*, 2017; SARKER *et al.*, 2020; SARKER *et al.*, 2018) reached promising results by predicting EC numbers based on the similarity between protein domains. The domain labels are obtained by analyzing the sequence with the three-dimensional structure bound to it, as shown in figure 1.

Protein domains are functional and structural units of a protein. In EC prediction algorithms, a binary feature vector is associated with each protein and the elements of the vector indicate the presence or absence of a domain. Thus, calculating the similarity between domain vectors of two proteins plays a fundamental role in many EC prediction methods.

2.2.1 Protein Function Annotation with Similarity Measures

Proteins are long sequences of amino acids that form the basis of life and plays a vital role in all living organism throughout the entire life-cycle. Proteins perform

Figure 1 – Protein Interpro Domain

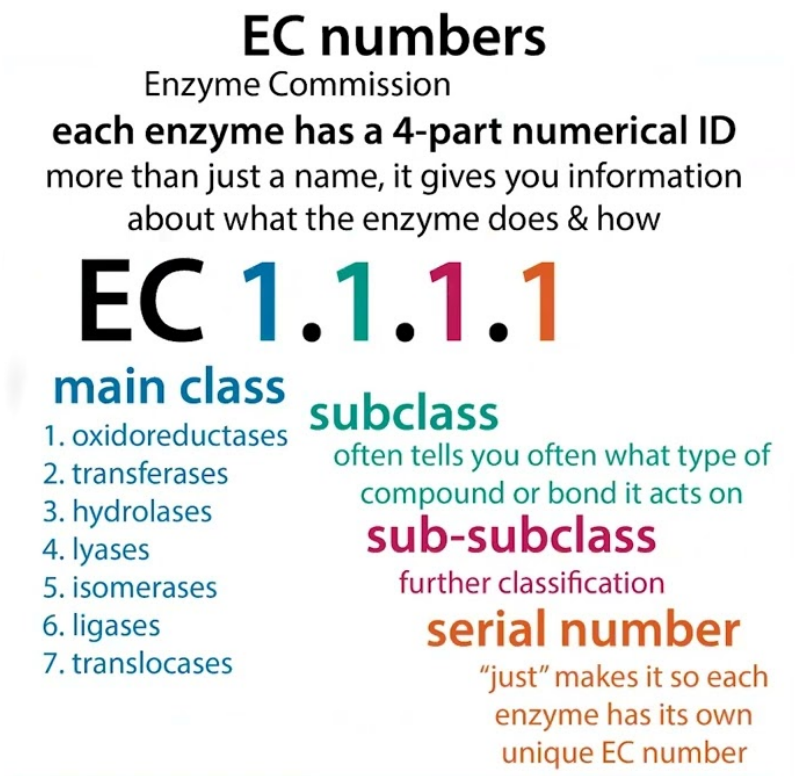


various functions in our body that needs to be understood to understand life, disease processes, and guiding drug discovery efforts to combat the diseases (BAKHEET; DOIG, 2009b). Due to the tremendous advancement in amino-acid sequencing technologies, it is now possible to sequence bulk amounts of proteins rapidly and affordably. Therefore, the number of protein sequences accumulating in public databases is rising at an unprecedented rate (BERGER *et al.*, 2016).

This huge quantity of data calls for further exploitation and enrichment and it presents many challenges for biologists as well as computer scientists in annotating the functional properties of protein sequences such as Enzyme Commission (EC) numbers, Gene Ontology (GO) Annotation, for example. The UniProt knowledge base (UniProtKB) (The UniProt Consortium, 2015) is currently the largest public sequence database. It consists of two components: (i) the UniProtKB/Swiss-Prot database which contains protein sequences with reliable information that has been reviewed by expert bio-curators, and (ii) the UniProtKB/TrEMBL database that stores unannotated sequences. Thus, for all proteins in UniProtKB, we have the primary amino acid sequences as well

as some further information such as InterPro domain composing the protein structure. Enzymes are usually labeled following the EC system (CORNISH-BOWDEN, 2014), the widely used numerical enzyme classification scheme. The EC System assigns each enzyme a four digits number. The EC classification system has a hierarchical structure. The first level consists of the six main enzyme classes: (i) oxidoreductases, (ii) transferases, (iii) hydrolases, (iv) lyases, (v) isomerases and (vi) ligases, represented by the first digit. Each main class node further extends out several subclass nodes, specifying subclasses of the enzymes, represented by the second digit. Similarly, the third digit indicates the sub-subclass and the fourth digit denotes the sub-sub-sub classes. Let us consider as an example a Type II restriction enzyme, which is annotated as EC 3.1.21.4. The first digit, 3, denotes that it is a hydrolase. The second digit, 1, indicates that it acts on ester bonds. The third digit, 21, shows that it is an endodeoxyribonuclease producing 5-phosphomonoesters. The last digit, 4, specifies that it is a Type II site-specific deoxyribonuclease. Therefore, the main objective of automatic EC number annotation is to automatically associate EC number to the query protein based on structural and sequence similarities. Figure 2 summarizes the EC number structure.

Figure 2 – Summary of EC numbers structure



2.3 Genetic Programming

Genetic Algorithm (GA) are evolutionary inspired learning algorithms. To select the best individuals to solve a problem, they use the Darwinian principle of reproduction and survival of the fittest and perform operations in the population that corresponds to genetic behaviors in nature, such as crossover and mutation. There are several features common to all GAs, such as designing each individual in a population as a possible solution to the problem, selecting the better-adapted individuals to pass on their traits to the next generation, applying genetic operators to the individuals before the next generation, and using a fitness-based selection method to measure which individuals have a higher probability of passing on their genes and thus increasing the likelihood that their genetic material will survive the entire process (ESPEJO *et al.*, 2010).

Genetic Programming (GP) is a direct variant of GAs in which individuals are different programs competing to accomplish a particular task. A typical GP's individual is represented by a tree composed of terminal and function nodes (BANZHAF *et al.*, 1998). Through the course of GP's execution the reproduction, mutation, crossover, and selection operations are performed each iteration over the population. The selection and reproduction intend to select the individuals and carry their genes, maintaining the best ones, while the mutation and crossover objective is to provide a variability factor (ROSA; PAPA, 2019). At the end of the execution, the best individual is chosen as the best solution found.

As described in (WILLIS *et al.*, 1997), the main differences between GAs and GPs are: GP typically has variable-length chromosomes, while GA generally uses fixed-length chromosomes. GAs are generally syntax-free, while GPs contain domain-specific syntax that decides what is a significant arrangement of information. The crossover and mutation of a GP must preserve meaning, i.e., the domain of syntax. Each chromosome of an individual in a GP problem could be executed as a standalone program with a suitable interpreter, but generally, GAs are not coded in this way

To apply genetic programming to a problem, there are five important preparatory steps(KOZA, 1995) :

The set of terminals - The terminals are the possible input variables and

constants for the computer program generated by the tree representation. They are located in the leaves of the trees.

The set of primitive functions - In the tree representation, anything that is not a leaf represents a primitive function. Functions should be able to accept as an argument any value contained in the terminal set or returned by any function in the function set.

The fitness measure - The fitness measure is a function that quantifies the quality of the output of the computer program for each individual of the GP process. The value that GP tries to achieve with its fitness function depends on the work to be done, e.g., it tries to minimize the error in a regression, minimize the consumption of a resource in an optimization, maximize the precision in a classification, etc.

The parameters to controlling the run - The population size and the probabilities of mutation and crossover are hyperparameters of GP. While the population size controls the maximum number of offspring, the probabilities determine how much the system tends to explore rather than exploit.

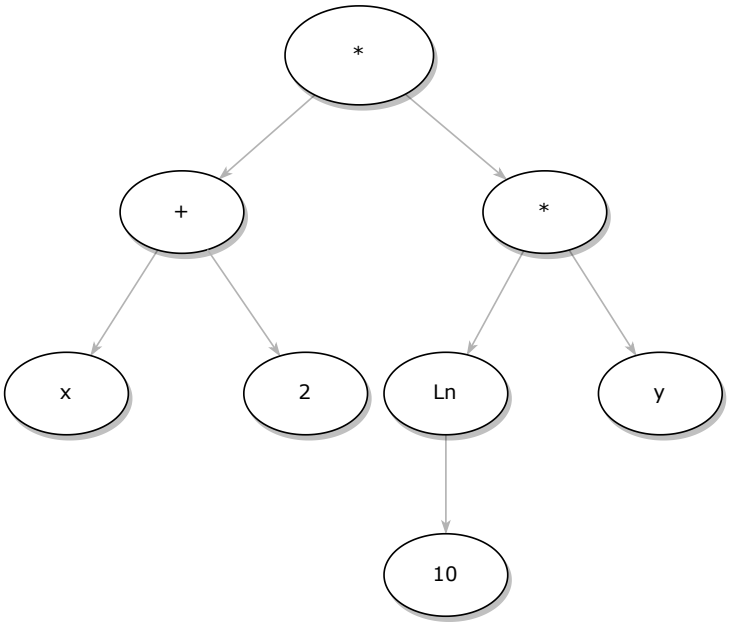
The stop criterion - Usually, the number of generations is defined as a criterion for the completion of the run, but it can also be a threshold for the fitness measure. After the stop criterion is satisfied, the tree with the best solution (better fitness) is selected.

For example, the figure 3 could represent an individual of a GP process with a set of terminals being x, y, z and a random integer between 1 and 10; the set of primitive functions being Ln (natural logarithm), * (multiplication), + (sum) and - (subtraction). It is evident that the terminal z and the subtraction function were absent from this particular individual, it does not mean that these values are eliminated from the entire process, however, other individuals could have them, or they could be acquired from a random mutation.

2.4 Conclusion

A brief review of the theoretical content of the Similarity measure and Binary Similarity Measure (BSM) are presented in this chapter, including its properties, principal

Figure 3 – Example of GP individual tree representation



One solution representing the equation $(x + 2)(ln(10)y)$

characteristics, and some examples. Furthermore, it was introduced to the problem of protein function annotation, how they are described as groups of amino acids, how those amino acids are transformed into binary vectors of domains, as well as how EC classification works. Further, it was briefly explained genetic programming (GP), which uses techniques to simulate Darwinian evolution to breed computer programs. We also described all the necessary steps and preliminary parameters before starting a GP process.

3 DESIGN OF A BINARY SIMILARITY FUNCTION FOR PROTEIN FUNCTION ANNOTATION

As already stated in section 2.2, predicting function based on protein sequences or protein structures is an important task. For instance, Sarker *et al.* (2020) was able to use Jaccard index to automatically annotate EC numbers. Although Jaccard is a well-known and good general function, it was not specifically designed for tasks of this nature. Due to the sparseness of data, shared domains tend to be quite small and a new BSM that takes that into account may be necessary.

As part of designing a new BSM for EC number prediction, we need to analyze how other functions perform in the same task as well as the characteristics of the dataset. Our approach is composed of three main steps: dataset building, design of a new similarity function, and similarity function evaluation.

Build a multi-labeled protein-protein dataset - A large number of state-of-the-art similarity measures are computed on a set of protein pairs. Each protein pair is labeled as sharing or not a function. To take maximal advantage of EC annotations, the shared function corresponds to one EC level. For example: If proteins a and b are both transferases then they share the first EC class level. If proteins c and d are both Type II site-specific deoxyribonuclease classes then they share the fourth EC class level. Each protein pair gets four labels corresponding to the four EC levels. Indeed a protein pair may share the same function at more than one level (e.g., Proteins annotated with 3.1.21.4 and 3.1.10.5 respectively share the same EC class at levels 1 and 2). The details on the dataset construction are provided in section 3.1. The multi-labeled dataset is split into two half-size subsets: the validation dataset and the evaluation (or test) dataset. These datasets will be used in the subsequent steps.

Design of a new similarity function - In this step, we aim to design a new measure based on empirical quantitative and qualitative analyses of the state-of-the-art similarity measures. These analyses are performed on the validation dataset. The aim is to select, based on their performance in the task in hand, the best state-of-the-art similarity measures. The qualitative analysis of their properties allows us to design our SSB measure. The procedure and the results of this step will be described in section 3.2.

Performance Evaluation - The last step comprises the evaluations of our

proposal on the evaluation/test dataset in comparison with other BSMs. The procedure and results will be presented in 3.4.

3.1 Construction of multi-labeled Protein-protein dataset

We selected a set of 63 similarity functions that were also used in Wijaya *et al.* (2016), Consonni and Todeschini (2012b) and Brusco *et al.* (2021). The experiment consists of randomly selecting pairs of proteins, calculating the similarity between their domains, represented by binary vectors, and using such similarities to decide whether the proteins share the same function. We used a set of 794 proteins, containing a total of 1132 domains and 394 unique EC numbers, that resulted in 314821 pairs of proteins. The proteins are retrieved from the UniProtKB/Swiss-Prot database (The UniProt Consortium, 2015), filtered by the Rattus norvegicus proteome, and selected those who had a valid EC number and at least one Interpro domain.

First, we split the set of proteins to build two datasets of the same size. The first dataset is used in the analysis and proposal of the new similarity function. The second set is used to evaluate the performance of the proposed function. After that, for each protein in each subset, we calculate the binary vector of presence/absence of domains, which results in a 396×1132 binary matrix. Being $\bar{X}$ the resulting matrix, the value of $\bar{X}_{i,k}$ for the protein i and the Interpro domain k is 1 if i had the domain k among its domains, and 0 otherwise.

For the final database X, we made a 2-combinations of rows in the previous data and, for each pair we create the four derived quantities a, b, c and d as described in 2.1.1. The quantities are used as input of the selected similarity functions, so, assuming $\mathbf{x}_i$ and $\mathbf{x}_j$ are 2 rows of $\bar{X}$, and $\mathbb{S}_k$ is the the k-th similarity function them, $\mathbb{S}_k(\mathbf{x}_i, \mathbf{x}_j)$ will be the k-th column of X with given similarity for each possible pair $\mathbf{x}_i$ and $\mathbf{x}_j$ with $\mathbf{x}_i \neq \mathbf{x}_j$.

Four different targets are computed, one for each level of EC number, the first level uses the first number as its class, the second level uses the first 2 numbers, since the second number is not uncorrelated from the first one, and we do the same for the next levels 3 and 4 using all the numbers before them. The target arrays Y are calculated similarly to X, but instead of applying a similarity function to a pair of proteins, we just observe if the class is the same in both. If this is true, the output value is 1, and 0 otherwise.

3.2 Design of a Similarity measure for Sparse Binary vectors (SSB)

To propose a new similarity function, we started by executing a set of experiments with well-known similarities for the EC prediction problem. Our main objective is to verify the characteristics of the best functions for the application. Such characteristics can be used to propose a new metric.

We used 63 similarity measures used in Wijaya *et al.* (2016), Consonni and Todeschini (2012b) and Brusco *et al.* (2021) and computed the similarities between pairs of proteins comprised in the validation dataset. The performance of each similarity measure is assessed with the Area Under the Precision-Recall curve (AP) metric.

For each EC level, we randomly sampled the dataset 50 times and computed all similarities and the AP for each measure in each run. After each execution we rank similarity functions as follows: for each run, the best performing algorithm gets the rank of 1, the second-best algorithm ranks 2, and so on. After computing the ranking for all 200 runs (50 repetitions and 4 EC levels) we calculated the average ranking of all similarity functions. As a result of such exploratory study, Yulew, Yuleq, Tarwid and Pearson Heron 1 were the Similarity functions with the best overall ranking.

By observing the best-performing metrics, one can notice that most of them have a numerator term that is close to $ad - bc$. Such a term is associated with the correlation between the two binary vectors. The main idea of $ad - bc$ is straightforward since it comprises a number of similarities (ad) minus a number related to dissimilarities (bc). Hence, we decided to include such term in our proposal. It's worth noting that the denominator is usually set as a normalizing factor, then we included the denominator term n^2 in our formulation.

One important practical aspect related to the application that shall be highlighted is the magnitude of the terms a, b, c, and d. Since we have a large number of domains and each protein has just a small amount of them, the d term is often much higher than the others. Hence, d is the main driver of the magnitude of $ad - bc$. Such behavior is undesirable since the shared domains must play an important role in the EC prediction problem. For that reason, we included a new factor, $e^{-\delta a}$, that depends on a. The exponential term is used to limit the influence of a between 0 and 1 so that the magnitude of this term is compatible with the rest of the expression. The impact of the new factor depends on the hyperparameter δ that may be set by any cross-validation

strategy. We also included a max operator with the ad term so that, in situations where the a term is high, its impact on the final measure is increased. The resulting similarity measure, denoted by $\mathbb{S}'$ is presented in Eq 3.1. The normalized version of our proposal is obtained by including $\mathbb{S}'$ in a logistic function. Thus, the final version of our proposal is shown in Eq. 3.2.

$$\mathbb{S}'(a,b,c,d) = \frac{\max(d,a)a - bc}{n^2} - e^{-\delta a} \tag{3.1}$$

$$\mathbb{S}_{SSB}(a,b,c,d) = \frac{1}{1+e^{-\lambda \mathbb{S}'(a,b,c,d)}} \tag{3.2}$$

Since a fundamental characteristic of the dataset is the sparseness of the domain vectors, the proposed method is termed Similarity Measure for Sparse Binary Vectors (SSB). The entire process of elaborating the proposed similarity function is summarized in Figure 4

3.3 Properties of SSB

To evaluate the properties of SSB, we can make use of the properties of Equations 3.1 and 3.2.

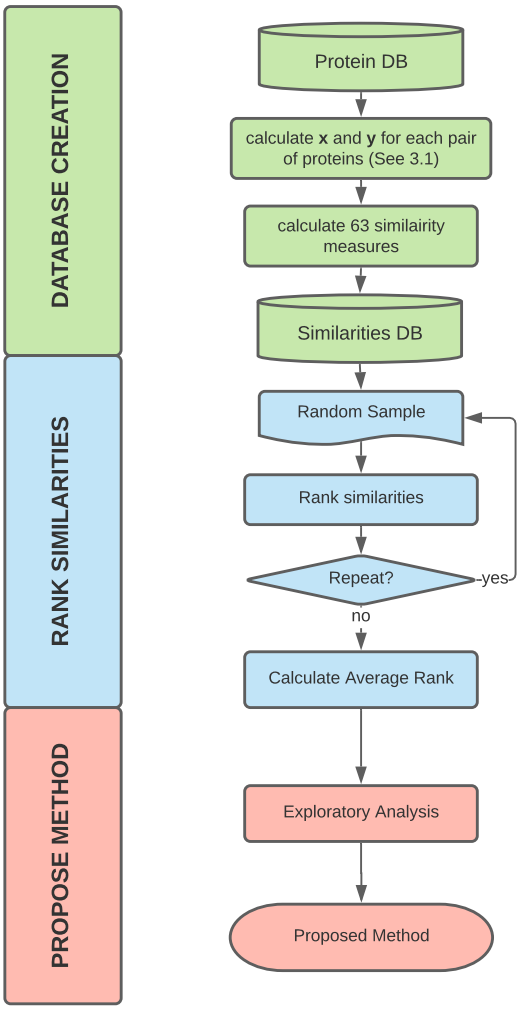
Non-negativity - Since Equation 3.2 is just the logistic function with parameters $\mathbb{S}'$ and λ, it has its limits between 0 and 1 and is a strictly increasing monotonic function, therefore, non-negativity is already verified.

Symmetry - Proving symmetry in Equation 3.1 is enough to prove that these properties are valid for Equation 3.2.

From the four quantities a, b, c, d that summarize the information of the binary vectors, the only ones that can change when we invert the vectors' input order are b and c (b becomes c and conversely). In Equation 3.1, the inversion of values between b and c does not cause any change in the result, since multiplication has the commutative property.

Maximality - Using the derived quantities and Equation 3.1 we can prove the maximality property as well. Being $\mathbf{x}$ and $\mathbf{y}$ two binary vectors and using the method introduced in 2.1.1 we will obtain a pair of derived quantities: $\bar{a}$, $\bar{b}$, $\bar{c}$, $\bar{d}$ the derived quantities from the pair $\mathbf{x}$, $\mathbf{y}$ and a, b, c, d from vector $\mathbf{x}$ by itself. To achieve the maximality, we need to show that: $\frac{\max(d,a)a-bc}{n^2} - e^{-\delta a} \geq \frac{\max(\bar{d},\bar{a})\bar{a}-\bar{b}\bar{c}}{n^2} - e^{-\delta \bar{a}}$. To make this

Figure 4 – SSB design methodology



task easier we can rewrite the equation as below:

$$\frac{\max(d,a)a - \max(\bar{d},\bar{a})\bar{a} + \bar{b}\bar{c} - bc}{n^2} \geq e^{-\delta a} - e^{-\delta \bar{a}} \tag{3.3}$$

The maximum number of positive matches of $\mathbf{x}$ and $\mathbf{y}$ is the number of positive values in $\mathbf{x}$, then $a \geq \bar{a}$ is true for any pair $(\mathbf{x}, \mathbf{y})$. One can conclude that the right-hand side of the equation $e^{-\delta a} - e^{-\delta \bar{a}}$ is less or equal than 0 by following the steps:

$$\begin{aligned} \delta a \geq \delta \bar{a} \quad &\Rightarrow \\ -\delta a \leq -\delta \bar{a} &\Rightarrow \\ e^{-\delta a} \leq e^{-\delta \bar{a}} &\Rightarrow \\ e^{-\delta a} - e^{-\delta \bar{a}} \leq 0 & \end{aligned}$$

For the left-hand side, the term bc is removed since they are both 0. The equation for proving maximality ends up being:

$$\frac{\max(d,a)a - \max(\bar{d},\bar{a})\bar{a} + \bar{b}\bar{c}}{n^2} \geq 0 \tag{3.4}$$

n^2 can be easily removed from the last equation since it is greater than 0. So, we can summarize Equation 3.4 to:

$$\max(d,a)a + \bar{b}\bar{c} \geq \max(\bar{d},\bar{a}) \tag{3.5}$$

From Equation 3.5 one of the following four scenarios may take place:

$$a^2 + \bar{b}\bar{c} \geq \bar{a}^2 \tag{3.6}$$

$$a^2 + \bar{b}\bar{c} \geq d\bar{a} \tag{3.7}$$

$$da + \bar{b}\bar{c} \geq \bar{a}^2 \tag{3.8}$$

$$da + \bar{b}\bar{c} \geq d\bar{a} \tag{3.9}$$

Equations 3.6 and 3.9 can be proved to be true from the fact that $\bar{b}\bar{c}$ is greater or equal than zero and $d \geq \bar{d}$ which can be derived from the same argument used to show $a \geq \bar{a}$ by inverting all boolean values in $\mathbf{x}$ and $\mathbf{y}$. Equation 3.7 is the scenario which $a \geq d$ and $\bar{d} \geq \bar{a}$, so we can conclude that $a \geq d \geq \bar{d} \geq \bar{a}$ that implies $a^2 \geq d\bar{a}$. The same can be applied in Equation 3.8, the scenario is $d \geq a$ and $\bar{a} \geq \bar{d}$ and the veracity of the equation is given by $d \geq a \geq \bar{a} \geq \bar{d}$ that implies $da \geq \bar{a}^2$.

3.4 Evaluation of SSB through two experiments

To assess the performance of SSB we conducted two experiments. The experiments aim to evaluate SSB on protein function annotation tasks. A third experiment, not related to protein function annotation but also generates a sparse dataset is additionally provided in the appendix B

3.4.1 Evaluation of SSB on protein function annotation task

The first protein function annotation experiment consists of the same pairwise comparison used in the method design phase (described in subsection 3.2) but with the test dataset. In the second experiment, we evaluate the performance of our similarity

metric as a component of the network-based protein function prediction algorithm proposed in (SARKER *et al.*, 2020). Such model computes the similarities between proteins and uses label propagation strategies to predict the functions. The original work uses the well-known Jaccard similarity.

For assessing the majority of the experiments we utilize the Average Precision (AP) (ZHU, 2004). The AP metric is one way of summarizing a precision-recall curve in a single value representing the average of all precisions. Being R the recall array of size n, P the precision array of size n, n the number of thresholds, and k the index that accesses the k-th element of a given array, The AP equation is given by:

$$AP = \sum_{k=0}^{n-1} \left(R_k - R_{k+1}\right) P_k \tag{3.10}$$

In other words, the equation 3.10 is derived from the difference between the current and next recalls, the difference is then multiplied by the current precision by using a loop that goes through all precisions/recalls.

3.4.1.1 Evaluation of SSB on the multi-labeled protein-protein dataset

Using the testing set, we computed the AP for all similarity measures and EC levels. Table 1 shows the APs for SSB, the best similarity at each EC level, and the Jaccard metric. The Jaccard similarity was chosen since it is one of the most popular binary similarity metrics. The statistical significance of the results is evaluated with the paired t-test. In such test, the null hypothesis states that the difference between SSB and each method is not significant.

Table 1 – Performance comparison of SSB, Jaccard and best similarity for each EC target

	EC 1	**EC 2**	**EC 3**	**EC 4**
SSB	0.6466 ± 0.0067	0.8359 ± 0.0113	0.7624 ± 0.0142	**0.2497 ± 0.0279**
JACCARD	0.5665 ± 0.0061 ✗	0.8122 ± 0.0125 ✗	0.7815 ± 0.0143 ✗	0.2065 ± 0.0234 ✗
BEST	**0.6510 ± 0.0071** ✓	**0.8423 ± 0.0101** ✓	**0.7820 ± 0.0148** ✗	0.2460 ± 0.0268 ✓

The best similarities from EC 1 to EC 4 are: Dennis, Pearson 2, Braun Banquet and Michael respectively. The symbols ✓ and ✗ indicate the result of the hypothesis test (✓ fail to reject, and ✗ reject). The best results are indicated in bold.

As can be noticed, SSB was able to outperform Jaccard in three out of four EC levels and the difference is statistically significant. It is also important to highlight

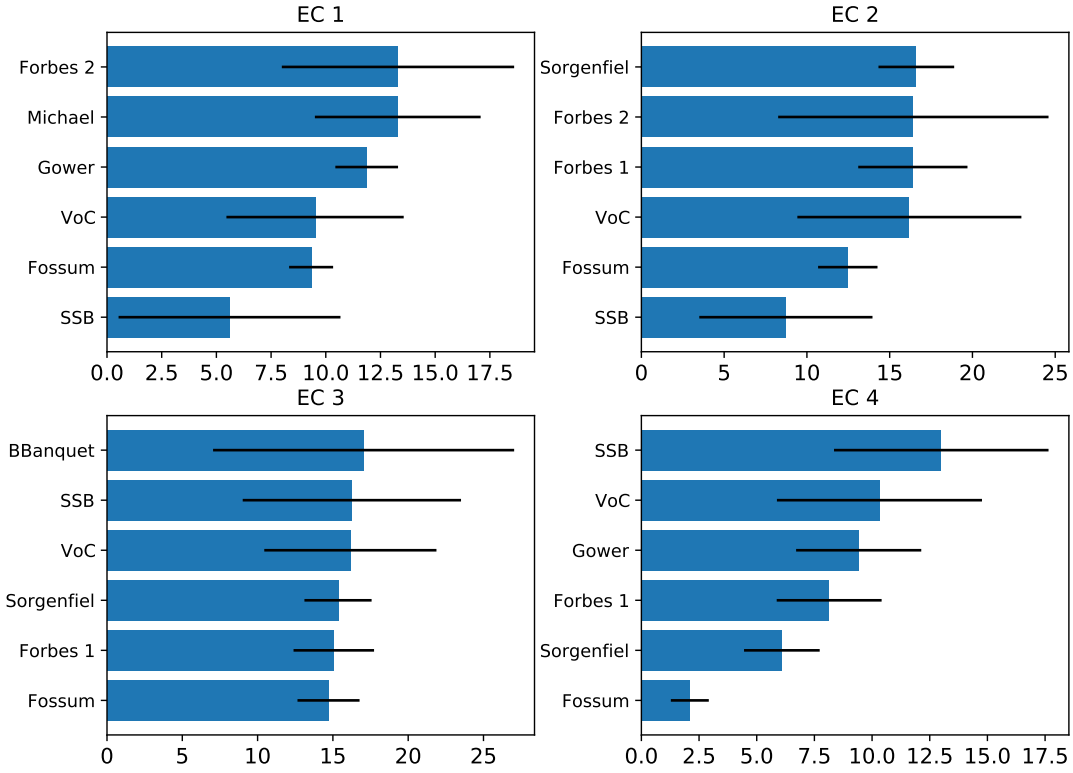
that there is no significant difference between SSB and the best similarity method for the same three out of four EC levels.

To derive another performance indicator, we decided to create a nonlinear similarity measure by combining all 63 measures. For that, we trained a random forest to predict whether a given pair of proteins share the same function and used all 63 similarities as numeric features. The random forest model was trained and evaluated using 10-fold cross-validation and achieved an mean AP of 0.657 ± 0.016, 0.85 ± 0.017, 0.788 ± 0.031 and 0.252 ± 0.036 for EC levels 1 to 4.

As the random forest achieved remarkable results, one interesting performance indicator is the importance of each feature used in the model. The average feature importance for each EC level is shown in Fig 5. Since all 63 similarities are used as features, being one of the most important similarities (according to the random forest model) is an indicator that SSB provided useful information to the classification task.

Once again SSB had a good performance and is among the six best similarities for all EC levels. Also, our measure had the best ranking in EC level 4 which is the most precise annotation level.

Figure 5 – Feature importance of SSB



Feature importance for the six best similarities in each EC level. The highest values indicate the most important features.

3.4.1.2 Evaluation of SSB in a label propagation process

In this experiment, we included SSB as part of a network-based protein function annotation method. In its original formulation, GrAPFI (SARKER *et al.*, 2020; SARKER *et al.*, 2018) used the Jaccard similarity function to build its protein network. By doing such an experiment, we aim to verify the impact of SSB in a recently proposed protein function annotation method. GrAPFI (SARKER *et al.*, 2020; SARKER *et al.*, 2018) is a neighborhood-based label propagation approach that works on a network of proteins connected using domains and family information. GrAPFI follows the following steps to perform function annotation:

First, it constructs a graph using the protein information. Each node u of the graph represents a protein. An edge (u, v) between two nodes/proteins u and v means that the linked proteins share some attributes, particularly Interpro domains in this case. A node u may have a set of labels $L(u)$ (one or more annotations to propagate), has a set of neighbors $N(u)$, and for every neighbor $v \in N(u)$, it has an associated weight $W_{u,v}$. GrAPFI uses Jaccard similarity to compute the link weight and computed as $W_{P1,P2} = \frac{|D1 \cap D2|}{|D1 \cup D2|}$ for two proteins P1 and P2 having sets of domains $D1 = d1, d2, d3$ and $D2 = d2, d3, d4$, respectively.

After that, a label propagation approach is applied to the protein graph to infer the functional properties of the unlabeled nodes. Given a query protein, based on the domains and family information, all the neighboring proteins and their annotations are retrieved from the weighted graph. After having the neighbors, each of the labels of the neighbors is weighted with edge-weights that these neighbors exhibit with this query protein. When retrieving neighbors, it is possible to select only those neighbors which meet a certain similarity threshold. That means that the links can be filtered based on a predefined cut-off weight. For each candidate annotation, GrAPFI provides a confidence score.

Table 2 shows the resulting coverage, precision, recall and F1 metrics when SSB or Jaccard is used in GrAPFI. The paired t-test was used to evaluate the statistical significance of the results.

As one can notice, for similar coverage values, SSB resulted in better performance metrics in all tested situations. The performance gap in all situations is statistically significant according to the paired t-test.

Table 2 – Performance comparison of SSB and Jaccard on GrAPFI

	Jaccard				SSB			
	EC 1	**EC 2**	**EC 3**	**EC 4**	**EC 1**	**EC 2**	**EC 3**	**EC 4**
Coverage	0.7746±0.04 ✓	0.7746±0.0401 ✓	0.7746±0.0401 ✓	0.7746±0.0401 ✓	0.7706±0.0482	0.7706±0.0482	0.7706±0.0482	0.7706±0.0482
Precision	0.8996 ±0.0740 ✗	0.7721±0.0859 ✗	0.7047±0.0717 ✗	0.5688±0.0611 ✗	**0.9449±0.0467**	**0.8813±0.0735**	**0.8368±0.0897**	**0.668±0.0823**
Recall	0.8742±0.0854 ✗	0.7782±0.0818 ✗	0.7257±0.0666 ✗	0.5739±0.0617 ✗	**0.9425±0.0614**	**0.8837±0.0728**	**0.8409±0.0924**	**0.67±0.0828**
F1	0.8777±0.0808 ✗	0.762±0.0849 ✗	0.7049±0.0693 ✗	0.5673±0.0588 ✗	**0.9387±0.0556**	**0.8730±0.0762**	**0.8329±0.0920**	**0.6643±0.0829**

The symbols ✓ and ✗ indicate the result of the hypothesis test (✓ fail to reject, and ✗ reject). The best results are indicated in bold.

3.5 Ablation Study

In this section, we conduct an ablation study to analyze the impact of the $e^{-\delta a}$ term in SSB. A detailed impact analysis of such term is important since no other similarity metric shows a similar term. Roughly speaking, the function of $e^{-\delta a}$ is to increase the influence of a in SSB. Since d is greater than a in sparse binary data scenarios and a is known to be important in our application, the exponential term may balance the numerical impact of a and d in SSB.

The ablation experiment consists of repeating the experiment described in section 4.1.1 but adding an SSB variant without the $e^{-\delta a}$ term. In the experiment, we perform a pairwise comparison between proteins that share or do not have the same function. For such problem, we compute the AP metric and also the average position in a performance ranking of all similarity metrics. All tests were repeated 100 times and the value of δ was adjusted using cross-validation. The performance of SSB and SSB* (without $e^{-\delta a}$) is shown in Table 3.

Table 3 – Ablation study of SSB.

		SSB	**SSB***
EC 1	AP	**0.6472±0.0091**	0.6393±0.0089 ✗
	RANK	**3.120 ±1.3126**	6.23±0.886 ✗
EC 2	AP	**0.8402±0.0121**	0.8401±0.012 ✓
	RANK	**4.320±2.3089**	4.85±2.3629 ✓
EC 3	AP	**0.7617±0.0214**	0.7531±0.0209 ✗
	RANK	**20.22±11.1732**	24.28±10.4089 ✗
EC 4	AP	**0.2574±0.0232**	0.2524±0.0235 ✓
	RANK	**2.23±1.8194**	3.56±4.5088 ✓

The symbols ✓ and ✗ indicate the result of the hypothesis test (✓ fail to reject, and ✗ reject). The best results are indicated in bold.

According to the results, we can verify that SSB outperformed SSB* in all EC levels and for all metrics, even though the performance gap in EC2 and EC4 was not

statistically significant. Such results indicate that using the exponential term may be beneficial in some applications. Since δ is an adjustable parameter, using an automatic way to find the best δ seems to be a good alternative in practical applications.

3.6 Conclusion

A new BSM is presented in this chapter for the problem of protein function annotation. This function was designed taking into consideration the evaluation of other BSMs and their properties, as well as properties intrinsic to the problem. It was shown the methodology on which the function was built and the way the dataset was created. Furthermore, we selected a set of experiments that assess the performance of the new function and compare its performance against other BSMs, showing that our proposal is a good BSM and promising general framework for supporting the development of new BSMs.

4 AUTOMATIC DESIGN OF BINARY SIMILARITY FUNCTIONS

The present approach aims to provide an automated framework that creates a BSM that respects the properties of non-negativity, symmetry, and maximality. The objective of the achieved function is to take two binary one-dimensional arrays and return a value that better represents the resemblance between the vectors in comparison to other similarities.

4.1 Building the BSM framework

To create an automated framework that produces BSMs, the following steps must be taken:

First, it is necessary to define a function $\mathcal{F}$, like all BSMs, $\mathcal{F}$ is a function that takes any two binary arrays $\mathbf{x}_i$ and $\mathbf{x}_j$ from the same domain $\mathcal{X}$ and uses it associated values to calculate the resemblance between the two elements, i.e., $\mathcal{F}(\mathbf{x}_i, \mathbf{x}_j) \rightarrow \mathbb{R}$ for all $\mathbf{x}_i, \mathbf{x}_j \in \mathcal{X}$. Besides that, $\mathcal{F}$ should always maintain the three desired theoretical properties at the same time that some of its parts must be altered to better suit a given problem. This can be resolved by making $\mathcal{F}$ an outer function that takes an inner function f as input and a static fragment that interacts with f. The static part must guarantee the properties, as well as f, must give the outer function some variation.

In second, we need to design the static part of $\mathcal{F}$ and each step below ensure that the function follows the indicated property regardless of the output value of f.

Symmetry - Being f a function that takes any two binary arrays $\mathbf{x}_i$ and $\mathbf{x}_j$ and return a real number y, we can easily define a function f_s that enclose f into its construction and have the symmetry property by adding $f(\mathbf{x}_i, \mathbf{x}_j)$ and $f(\mathbf{x}_j, \mathbf{x}_i)$, since addition has the commutative property, i.e:

$$f_s(\mathbf{x}_i, \mathbf{x}_j) = f(\mathbf{x}_i, \mathbf{x}_j) + f(\mathbf{x}_j, \mathbf{x}_i) \tag{4.1}$$

Non-negativity - Now, combining f_s and the logistic function (COX, 1958), we can create a new function f_{nn}, that has the property of non-negativity in the following way:

$$f_{nn}(\mathbf{x}_i, \mathbf{x}_j) = \frac{1}{1 + e^{-f_s(\mathbf{x}_i, \mathbf{x}_j)}} \tag{4.2}$$

The logistic function has its limits between 0 and 1 and is a strictly increasing monotonic function, therefore, non-negativity is already verified.

Maximality - The maximum value of the logistic function is achieved when the denominator is minimum then, we can introduce a term in the denominator of f_{nn} that multiply the exponential function making it zero every time $\mathbf{x}_i = \mathbf{x}_j$. Using the associated values in 2.1.1 and according to their definitions, b and c are both zero if $x = y$. Then, the final the equation is given by:

$$\mathcal{F}(\mathbf{x}_i, \mathbf{x}_j) = \frac{1}{1 + sign(b + c)e^{-(f_s(\mathbf{x}_i, \mathbf{x}_j))}} \tag{4.3}$$

where,

$$sign(z) = \begin{cases} 1, & \text{if } z > 1 \\ 0, & \text{otherwise} \end{cases} \tag{4.4}$$

for any $z \in \mathbb{Z}^+$

Though the proposed framework above intends to create a function $\mathcal{F}$ that has three properties above for any inner function f. However, it is unlikely that a random f would make $\mathcal{F}$ into the most suitable function for a given problem. As such, it is necessary to use a strategy that performs a function approximation in $\mathcal{F}$ but modifies only f. And since GP is an effective tool for this purpose(SHARIATI *et al.*, 2019), the next section will describe how we approach the five preparation steps contained in 2.3.

4.2 Defining GP's components

In the creation of the defined f function by the use of GP, we need to set the GP's components, which are: the set of terminals, The set of primitive functions, the fitness measure, the parameters to control the run and the stop criterion.

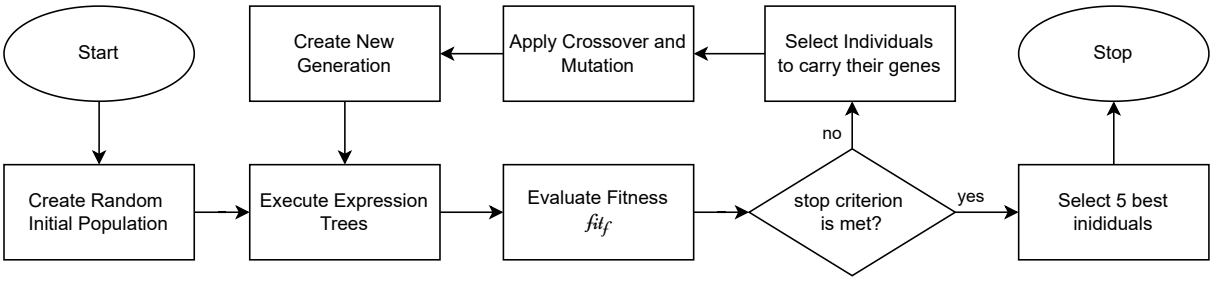
First, we have the set of terminals, which in this case are: the associated values defined in 2.1.1, the sum of the associated values (n), and the constant values $-1, 1$ and 2. Secondly, to keep things simple, we used only four primitive functions: addition, subtraction, multiplication, and division, with a trick that ensured that division by zero would return 1 instead of an undefined value.

During the third step, we define how the fitness measure is defined. Although the GP's tree represents f, the ultimate goal is for $\mathcal{F}$ to have a higher AP, since it

contains all the properties we need. Being f an individual of the GP process, $\mathcal{F}_f$ the function defined in equation 4.3 in relation to f, X the set of inputs of the training data, $\mathbf{y}$ the set of targets of the training data and AP_f the average precision of $\mathcal{F}_f$ then, we can define the fitness function as $fit_f = 1 - AP_f$.

In the fourth step, we determine the size of the initial population and the probability that the selected individuals will mate and mutate. We selected 50 individuals for the initial population, and 0.3 and 0.05, respectively, for mate and mutation. Additionally, we add a maximum depth parameter to the tree representation, allowing the function to become more complex as it increases. For example, if a function has a maximum depth of 4 it can accommodate no more than 7 operators and 8 inputs/constants, while with a depth 8, the function can support as many as 127 operators and 128 inputs/constants. Our final step was to stop the process once we reached 1000 generations and then select the five best results out of all the results, i.e., the function with the highest AP over the training data.

Figure 6 – GP Flowchart



4.3 Evaluating the Framework over two datasets

We conducted two experiments to evaluate the performance of the proposed framework. In the first one, we are working on the annotation of protein functions, and in the second, we will investigate the relevance of the framework in a completely different task, co-authorship prediction.

4.3.1 Evaluation on the multi-labeled protein-protein dataset

The multi-labeled protein-protein dataset that was detailed described in subsection 3.1 was once again used to conduct our experiments. However, instead of

splitting similarly, we split the data set into three parts: training, validation, and test, with each involving 50%, 20%, and 30% of the data, respectively.

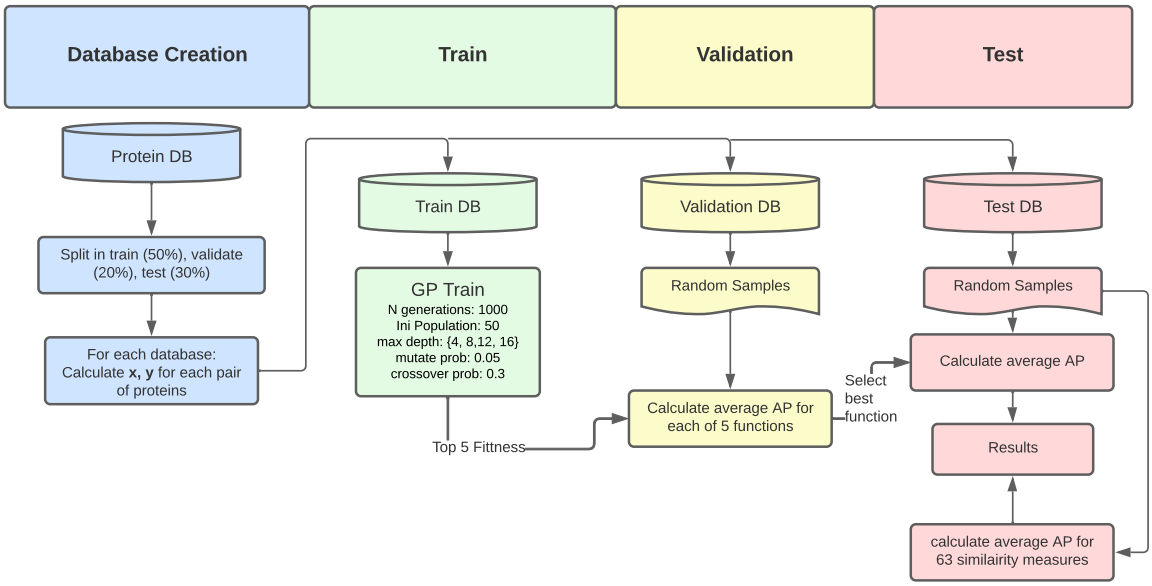
Training - In the training data, the GP environment described in subsection 4.2 will be used to create a set of similarity functions, from which 5 best ones will be chosen by their fitness function, that is, the BSMs with the highest AP.

Validating - To provide an unbiased evaluation for selecting one of the five resulting functions, we used the validation data, a slice of the data never seen in the training step. We spliced the validation data into 20 equal-sized subsets of rows and then calculated the AP for each subset for each function. The function with the highest mean AP is selected for evaluation against other similarity functions during the testing phase.

Testing - In the portion that will be used to test the data, we again divided the data into 20 equal-sized subsets and calculated the mean and the standard deviation for the selected function as well as the 63 similarity functions that were also used in Wijaya *et al.* (2016), and then used the paired t-test to determine if there is a significant difference between the chosen function and the other similarity functions.

The flowchart in Figure 7 summarize all the steps mentioned above.

Figure 7 – Flowchart with training, validation and test steps



The comparisons are provided in the table 4, each column represents an EC level, and each row shows the results of a selected function produced by genetic programming with a maximum depth of 4, 8, 12, and 16, the Jaccard similarity, and the best similarity among the 63 similarity functions computed. Jaccard similarity was chosen since it is one of the most widely used binary similarity metrics. The symbols ✗ and ✓ inside the parenthesis indicate the results of the hypothesis test, rejection and failure to reject, respectively. They are grouped in pairs, the first of which is a comparison with Jaccard, the second with the best similarity. One can see from the results that the framework's similarity function outperforms at all EC levels and that creating a less complex function is not statistically different from the best function, but as complexity increases, it starts to be. A very complex function, however, starts to overfit the training data and could degrade its accuracy in the test data.

Table 4 – Protein Function Annotation

	EC 1	**EC 2**	**EC 3**	**EC 4**
GP Depth 4	0.6639 ± 0.0083 (✗, ✗)	0.8536 ± 0.0132(✗, ✓)	0.7815 ± 0.0082(✓, ✓)	0.249 ± 0.0113(✗, ✓)
GP Depth 8	**0.6654** ± **0.0052**(✗, ✗)	0.8506 ± 0.0095 (✗, ✓)	0.8040 ± 0.0164(✗, ✗)	0.2662 ± 0.0141(✗, ✗)
GP Depth 12	0.6651 ± 0.006(✗, ✗)	**0.8583** ± **0.01** (✗, ✗)	0.8038 ±0.0116 (✗, ✗)	0.2589 ± 0.0169 (✗, ✓)
GP Depth 16	0.6652 ± 0.0053 (✗, ✗)	0.8548 ± 0.0059 (✗, ✗)	**0.8118** ± **0.028**(✗, ✗)	**0.2676** ± **0.0097** (✗, ✗)
JACCARD	0.5660 ± 0.0043	0.8117 ± 0.0092	0.7714 ± 0.0253	0.2046 ± 0.0125
BEST	0.6442 ± 0.0059	0.8421 ± 0.007	0.7718 ± 0.0256	0.2437 ± 0.0142

In EC levels 1 to 4, Var of Correlation, Pearson 1, Pearson Heron 1, and Dennis were selected as the best similarity functions respectively.

4.3.2 Evaluation of the framework on co-authorship prediction task

We will assess here the relevance/interest of the automatic framework that we design on a distinct task: predict the author of a review based on the occurring words. This task is important in several applications such as plagiarism detection, anonymity lifting, or document metadata completion.

Our experiment used data derived from: "Amazon Commerce Website for authorship identification"(DUA; GRAFF, 2017). In the original data, there are 50 authors and 10000 attributes(words) in 30 reviews from each author. Our objective is to predict if two reviews are from the same author based on the word comprised in the text.

To create the dataset, we filtered the most common words, remaining just the attributes that were contained in less than 1% of the comments. Then we computed a, b, c, and d for all pairs of reviews. If a pair or reviews are from the same author we

set the target as 1 and 0 otherwise. By analyzing the words in the text, we try to predict whether two reviews are written by the same author.

As with the previous problem this problem was also trained, validated, and tested similarly and the results can be found in table 5. As in the first evaluation, the framework outperformed Jaccard, and paired t-tests reject the hypothesis that it is statistically the same as well. However, even if the framework's mean AP is numerically superior, it cannot prove the difference between it and the best function.

Table 5 – Book Reviews

GP Depth 4	0.1191 ± 0.0003 (✗, ✓)
GP Depth 8	0.1201 ± 0.0003 (✗, ✓)
GP Depth 12	0.1208 ± 0.0003 (✗, ✓)
GP Depth 16	$\mathbf{0.1216 \pm 0.0002}$ (✗, ✓)
JACCARD	0.0967 ± 0.0002
BEST (Pearson 2)	0.1143 ± 0.0003

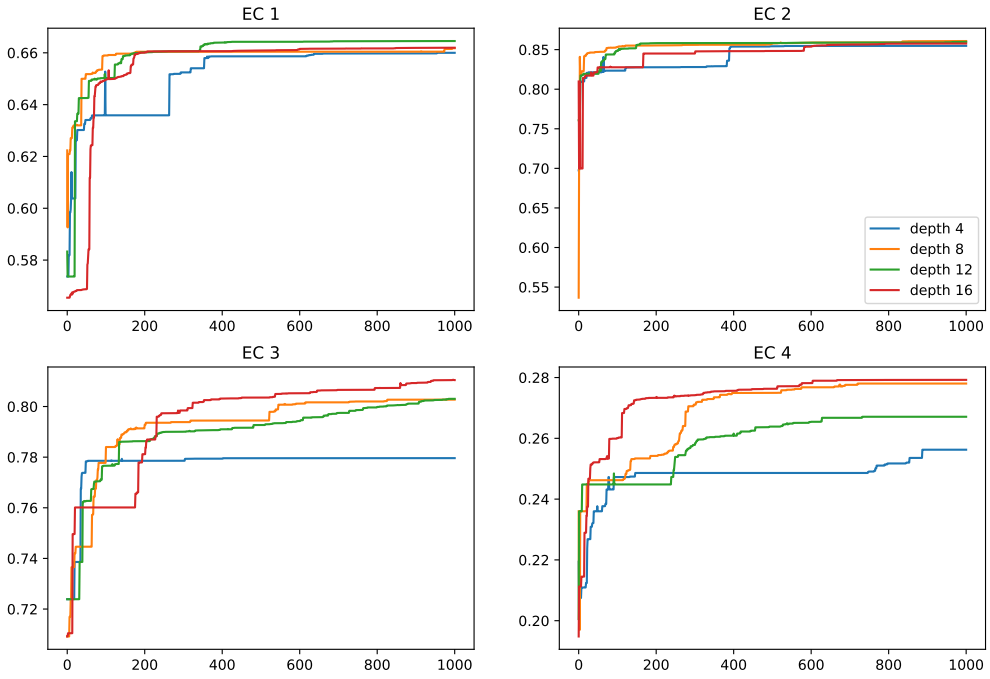
4.3.3 GP convergence Analysis

An important analysis to be done is the convergence of the GP. Often, populations stabilize after some time, with all the top fitness programs having a common ancestor and exhibiting behavior that is similar or identical to early generations programs. Without a proper mechanism to avoid early convergences, like mutation, the population can stuck in a solution not as high quality as expected. However, an early halt to the process may limit how high the metric can become.

As illustrated in Figure 8, one can see that the convergence of the EC 1 and EC 2 occurs earlier than that of EC 3 and EC 4, which is due to the complexity that grows with the dealt level, so the GP takes longer to converge. The convergence of populations allowed to have greater depths also took longer than the smaller ones, the complexity of the equation grows with the number of maximum possible nodes, which is exponential. However, is clear that the program is not finishing prematurely, as we can see the convergence over the iterations.

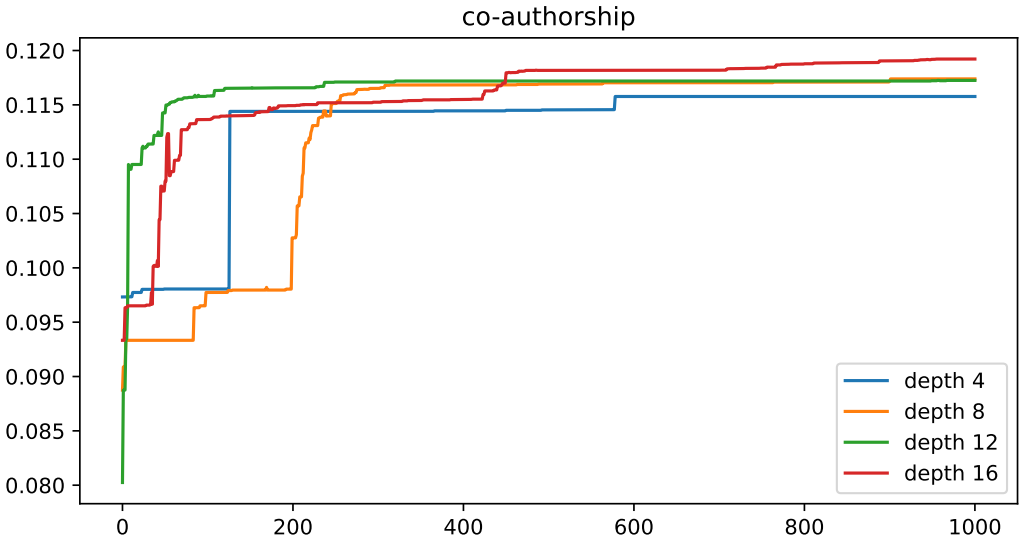
In figure 9, we can see the GP convergence graph in the co-authorship task. As the first convergence graph, the process was not finished prematurely, as the convergence occur, and in the final solution, the fitness of the population allowed to

Figure 8 – GP convergence in EC number annotation task



have deeper trees are higher than the shallower trees.

Figure 9 – GP convergence in co-authorship task



4.4 Conclusion

In this chapter, we presented an automatic framework to build BSMs. This framework makes use of encapsulation function to provide a way of maintaining the properties o symmetry, non-negativity, and maximality, at the same time that it learns from data using a function-approximation method. We showed how to use GP in the context of the framework, using the derived values of binary representation as inputs. Furthermore, we evaluated the automatic framework in the context of two real-world scenarios: the same function annotation problem explained in chapter 3 and the co-authorship prediction. We also made an auxiliary convergence analysis to make sure the training step is converging. The relevance of the results showed that the framework was able to surpass the other off-the-shelf BSMs.

5 CONCLUSION

In the first part of this work, we introduce a new similarity function for binary data designed for the annotation of protein functions. The design of the new function, named Similarity measure for Sparse Binary vectors (SSB) is based on the best performing similarity measures for a protein function annotation task.

In our experiments, we evaluated SSB's capability to identify similarities between domains of proteins that share the same functions and its potential to improve the performance of a network-based protein annotation method. Although SSB was not specifically designed for other tasks, we included an experiment on a co-authorship prediction task in appendix B. Its performance in such a task indicates that SSB may be a valid alternative for other problems that involve similarity calculations on sparse binary data. It is important to point out that our proposal includes two hyperparameters, λ, and δ, that control the slope of the logistic function and the impact of a in SSB, respectively. Including such parameters can make SSB more flexible but can also add a parameter tuning stage to the deployment of our method. In that case, the tuning should be performed on a separate validation set. Also, the use of SSB in protein function annotation was only evaluated in EC prediction and although our method had good results, it is not possible to infer its performance in a more challenging task like GO terms prediction.

In our second proposal, we presented an automatic framework that is capable of generating new BSMs. In the process of making the framework, a data-driven methodology is used to create an inner function and encapsulate it into an outer function that follows all three similarity function properties: symmetry, non-negativity, and maximality.

The proposed framework was compared with off-the-shelf similarity functions in two experiments, demonstrating its ability to properly annotate protein functions and to predict co-authorship based on a shared group of words. We showed that any function approximation technique can be applied to build a similarity function in this framework, and by making use of genetic programming we were able to achieve better mean APs than other tested BSMs. The mean AP was the main metric used to assess the performance of the framework in two different tasks (EC level prediction and co-authorship prediction) and the experimental results show that the framework

outperforms the other similarities. In our experiments we just make use of depth as a variable hyperparameter for our GP's training, in a future approach a hyperparameter optimization stage could be incorporated in the training phase.

An overview of both proposals be found below:

- a methodology to design similarity functions for protein function annotation
- a new similarity measure, named Similarity measure for Sparse Binary vectors (SSB), that was designed for EC number prediction
- an empirical evaluation of SSB
- a methodology to create a framework capable of creating BSMs
- an automated framework that generates BSMs with the three desired properties
- an evaluation of the automated framework in two tasks (EC level prediction and co-authorship prediction)

Based on our results, we can state that SSB is a promising similarity measure for protein function annotation and may be a valid alternative for other applications. As stated earlier, in the current work SBB is evaluated only for EC prediction. However, the promise of SSB is not tested for predicting other types of functional characteristics of proteins, for example, GO term prediction - a relatively more challenging task. Various factors contribute to the complexity of GO term prediction including complex multi-layer hierarchical relations among the terms. However, a significant amount of work deals with protein-protein-interaction networks for GO term prediction where similarity measure plays a vital role. A potential future endeavor might be to analyze the efficacy of SBB with the necessary extension to perform automatic GO annotation tasks. Other potential applications could be ligand-binding site prediction, drug-target prediction, drug repurposing, and biomedical knowledge discovery. We also intend to use SSB to extend similarity-based methods in domains such as machine learning (HAMALAINEN *et al.*, 2020), decision under uncertainty (ZHANG, 2016; WANG; GARG, 2020) and computer vision (EELBODE *et al.*, 2020). We can also state, that the resulting framework is a promising approach for automating the creation of similarity functions since it outperformed the other 63 similarity functions that were used extensively in the literature. The table containing all the compared similarity functions and their formulas can be found in appendix A.

REFERENCES

ABREU, R.; ZOETEWEIJ, P.; GEMUND, A. J. V. An evaluation of similarity coefficients for software fault localization. In: **2006 12th Pacific Rim International Symposium on Dependable Computing (PRDC'06)**. [S.l.: s.n.], 2006. p. 39–46.

ALBORZI, S. Z.; DEVIGNES, M.-D.; RITCHIE, D. W. Ecdomainminer: discovering hidden associations between enzyme commission numbers and pfam domains. **BMC Bioinformatics**, BioMed Central, v. 18, n. 1, p. 107, Dec. 2017.

BAKHEET, T.; DOIG, A. Properties and identification of human protein drug targets. **Bioinformatics**, v. 25, p. 451–7, 02 2009.

BAKHEET, T. M.; DOIG, A. J. Properties and identification of human protein drug targets. **Bioinformatics**, v. 25, n. 4, p. 451–457, 2009.

BANZHAF, W.; FRANCONE, F. D.; KELLER, R. E.; NORDIN, P. **Genetic Programming: An Introduction: On the Automatic Evolution of Computer Programs and Its Applications**. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1998. ISBN 155860510X.

BELOHLÁVEK, R.; GRISSA, D.; GUILLAUME, S.; NGUIFO, E. M.; OUTRATA, J. Boolean factors as a means of clustering of interestingness measures of association rules. **Annals of Mathematics and Artificial Intelligence**, v. 70, n. 1-2, p. 151–184, 2014.

BERGER, B.; DANIELS, N. M.; YU, Y. W. Computational biology in the 21st century: Scaling with compressive algorithms. **Communications of the ACM**, ACM, New York, NY, USA, v. 59, n. 8, p. 72–80, Jul. 2016. ISSN 0001-0782.

BRUSCO, M.; CRADIT, D.; STEINLEY, D. A comparison of 71 binary similarity coefficients: The effect of base rates. **PLOS ONE**, v. 16, p. e0247751, 04 2021.

CHOI, S.; CHA, S.; TAPPERT, C. C. A survey of binary similarity and distance measures. **Journal of Systemics, Cybernetics and Informatics**, v. 8, n. 1, p. 43–48, 2010.

CONSONNI, V.; TODESCHINI, R. New similarity coefficients for binary data. **MATCH Communications in Mathematical and in Computer Chemistry**, v. 68, p. 581–592, 01 2012.

CONSONNI, V.; TODESCHINI, R. New similarity coefficients for binary data. **MATCH Communications in Mathematical and in Computer Chemistry**, v. 68, p. 581–592, 01 2012.

CORNISH-BOWDEN, A. Current IUBMB recommendations on enzyme nomenclature and kinetics. **Perspectives in Science**, Elsevier, v. 1, n. 1-6, p. 74–87, 2014.

COX, D. R. The regression analysis of binary sequences. **Journal of the Royal Statistical Society: Series B (Methodological)**, Wiley Online Library, v. 20, n. 2, p. 215–232, 1958.

DUA, D.; GRAFF, C. **UCI Machine Learning Repository**. 2017. Accessed: 2021-02-1.

EELBODE, T.; BERTELS, J.; BERMAN, M.; VANDERMEULEN, D.; MAES, F.; BISSCHOPS, R.; BLASCHKO, M. B. Optimization for medical image segmentation: Theory and practice when evaluating with dice score or jaccard index. **IEEE Transactions on Medical Imaging**, v. 39, n. 11, p. 3679–3690, 2020.

ESPEJO, P. G.; VENTURA, S.; HERRERA, F. A survey on the application of genetic programming to classification. **IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)**, v. 40, n. 2, p. 121–144, 2010.

HAMALAINEN, J.; ALENCAR, A. S. C.; KARKKAINEN, T.; MATTOS, C. L. C.; JUNIOR, A. H. S.; GOMES, J. P. P. Minimal learning machine: Theoretical results and clustering-based reference point selection. **Journal of Machine Learning Research**, v. 21, n. 239, p. 1–29, 2020.

HUBáLEK, Z. Coefficients of association and similarity, based on binary (presence-absence) data: An evaluation. **Biological Reviews**, v. 57, p. 669 – 689, 01 2008.

KABIR, S.; WAGNER, C.; HAVENS, T. C.; ANDERSON, D. T.; AICKELIN, U. Novel similarity measure for interval-valued data based on overlapping ratio. In: **2017 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE)**. [S.l.: s.n.], 2017. p. 1–6.

KOZA, J. R. Survey of genetic algorithms and genetic programming. In: **Proceedings of WESCON'95**. [S.l.: s.n.], 1995. p. 589–.

LESOT, M.-J.; RIFQI, M.; BENHADDA, H. Similarity measures for binary and numerical data: A survey. **International Journal of Knowledge Engineering and Soft Data Paradigms**, v. 1, p. 63–84, 01 2009.

LI, Y.; WANG, S.; UMAROV, R.; XIE, B.; FAN, M.; LI, L.; GAO, X. DEEPre: sequence-based enzyme EC number prediction by deep learning. **Bioinformatics**, v. 34, n. 5, p. 760–769, 10 2017. ISSN 1367-4803.

NAGARAJA, A.; BOREGOWDA, U.; KHATATNEH, K.; VANGIPURAM, R.; NUVVUSETTY, R.; KIRAN, V. S. Similarity based feature transformation for network anomaly detection. **IEEE Access**, v. 8, p. 39184–39196, 2020.

NASEEM, R.; DERIS, M. M. A new binary similarity measure based on integration of the strengths of existing measures: Application to software clustering. In: HERAWAN, T.; GHAZALI, R.; NAWI, N. M.; DERIS, M. M. (Ed.). **Recent Advances on Soft Computing and Data Mining**. Cham: Springer International Publishing, 2017. p. 304–315. ISBN 978-3-319-51281-5.

OLIVEIRA, S. A.; ALVES, S. S.; GOMES, J. P.; Rocha Neto, A. R. A bi-directional evaluation-based approach for image retargeting quality assessment. **Computer Vision and Image Understanding**, v. 168, p. 172 – 181, 2018. ISSN 1077-3142. Special Issue on Vision and Computational Photography and Graphics.

RÁCZ, A.; ANDRIĆ, F. L.; BAJUSZ, D.; HÉBERGER, K. Binary similarity measures for fingerprint analysis of qualitative metabolomic profiles. **Metabolomics**, v. 14, 2018.

ROSA, G. de; PAPA, J. Soft-tempering deep belief networks parameters through genetic programming. v. 1, p. 43–59, 07 2019.

SARKER, B.; KHARE, N.; DEVIGNES, M.-D.; ARIDHI, S. Graph based automatic protein function annotation improved by semantic similarity. In: ROJAS, I.; VALENZUELA, O.; ROJAS, F.; HERRERA, L. J.; ORTUÑO, F. (Ed.). **Bioinformatics and Biomedical Engineering**. Cham: Springer International Publishing, 2020. p. 261–272. ISBN 978-3-030-45385-5.

SARKER, B.; RITCHIE, D.; ARIDHI, S. Exploiting complex protein domain networks for protein function annotation. In: **Complex Networks**. [S.l.: s.n.], 2018.

SARKER, B.; RITCHIE, D.; ARIDHI, S. Grapfi: predicting enzymatic function of proteins from domain similarity graphs. **BMC Bioinformatics**, v. 21, 12 2020.

SHARIATI, M.; MAFIPOUR, M.; MEHRABI, P.; ZANDI, Y.; DEHGHANI, D.; BAHADORI, A.; SHARIATI, A.; NGUYEN-THOI, T.; SALIH, M.; SHEK, P. N. Application of extreme learning machine (elm) and genetic programming (gp) to design steel-concrete composite floor systems at elevated temperatures. **Steel and Composite Structures**, v. 33, p. 319–332, 11 2019.

SHEN, H.-B.; CHOU, K.-C. Ezypred: A top-down approach for predicting enzyme functional classes and subclasses. **Biochemical and biophysical research communications**, v. 364, p. 53–9, 12 2007.

SMARANDACHE, F.; COLHON, M.; VLăDUţESCU Ştefan; NEGREA, X. Word-level neutrosophic sentiment similarity. **Applied Soft Computing**, v. 80, p. 167 – 176, 2019. ISSN 1568-4946.

The UniProt Consortium. UniProt: a hub for protein information. **Nucleic Acids Research**, v. 43, n. D204-D212, Jan. 2015. ISSN 0305-1048.

TODESCHINI, R.; CONSONNI, V.; XIANG, H.; HOLLIDAY, J. D.; BUSCEMA, P. M.; WILLETT, P. Similarity coefficients for binary chemoinformatics data: Overview and extended comparison using simulated and real data sets. **Journal of Chemical Information and Modeling**, v. 52, n. 11, p. 2884–2901, 2012.

WANG, L.; GARG, H. Algorithm for multiple attribute decision-making with interactive archimedean norm operations under pythagorean fuzzy uncertainty. **International Journal of Computational Intelligence Systems**, v. 14, p. 503–527, 2020. ISSN 1875-6883.

WIDLAK, W. Protein structure and function. In: ____. **Molecular Biology: Not Only for Bioinformaticians**. Berlin, Heidelberg: Springer Berlin Heidelberg, 2013. p. 15–29. ISBN 978-3-642-45361-8.

WIJAYA, S.; AFENDI, F.; BATUBARA, I.; DARUSMAN, L.; AMIN, A.; KANAYA, S. Finding an appropriate equation to measure similarity between binary vectors: Case studies on indonesian and japanese herbal medicines. **BMC Bioinformatics**, v. 17, p. 520, 12 2016.

WILLIS, M.-J.; HIDEN, H.; MARENBACH, P.; MCKAY, B.; MONTAGUE, G. Genetic programming: an introduction and survey of applications. In: **Second International Conference On Genetic Algorithms In Engineering Systems: Innovations And Applications**. [S.l.: s.n.], 1997. p. 314–319.

WOLF, L.; HASSNER, T.; TAIGMAN, Y. The one-shot similarity kernel. In: **2009 IEEE 12th International Conference on Computer Vision**. [S.l.: s.n.], 2009. p. 897–902.

Wolpert, D. H.; Macready, W. G. No free lunch theorems for optimization. **IEEE Transactions on Evolutionary Computation**, v. 1, n. 1, p. 67–82, 1997.

YANG, J.; YAN, R.; ROY, A.; XU, D.; POISSON, J.; ZHANG, Y. The i-tasser suite: Protein structure and function prediction. **Nature methods**, v. 12, p. 7–8, 12 2014.

ZHANG, X. A novel approach based on similarity measure for pythagorean fuzzy multiple criteria group decision making. **International Journal of Intelligent Systems**, v. 31, n. 6, p. 593–611, 2016.

ZHU, M. Recall, precision and average precision. **Department of Statistics and Actuarial Science**, v. 2, 2004.

APPENDIX A – SIMILARITY FUNCTIONS

List containing all similarity functions used in the experiments

Ample $= |\frac{a*(a+b)}{c*(c+d)}|$

Anderberg $= \frac{\sigma-\sigma'}{2n}$

Baroni UrbaniBuser 1 $= \frac{\sqrt{ad}+a}{\sqrt{ad}+a+b+c}$

Baroni UrbaniBuser 2 $= \frac{\sqrt{ad}+a-(b+c)}{\sqrt{ad}+a+b+c}$

BraunBanquet $= \frac{a}{\max((a+b),(a+c)}$

Cosine $= \frac{a}{\sqrt{(a+b)(a+c)}}$

Dennis $= \frac{ad-bc}{\sqrt{n(a+b)(a+c)}}$

Derived Jaccard $= \frac{\log(1+a)}{\log(1+a+b+c)}$

Derived Rusell-Rao $= \frac{\log(1+a)}{\log(1+n)}$

Derived SokalMichener $= \frac{\log(1+a+d)}{\log(1+a+d)}$

Derived Log-SokalMichener $= \frac{\log(1+n)-\log(1+b+c)}{\log(1+n)}$

Dice 1/Czekanowski $= \frac{2a}{2a+b+c}$

Dice 2 $= \frac{a}{2a+b+c}$

Disperson $= \frac{a*d-b*c}{(a*d+b*c)^2}$

DriverKroeber $= \frac{a}{2}\left(\frac{1}{a+b}+\frac{1}{a+c}\right)$

Eyraud $= \frac{n^2(na-(a+b)(a+c))}{(a+b)(a+c)(b+d)(c+d)}$

FagerMcGowan $= \frac{a}{\sqrt{(a+b)(a+c)}} - \frac{\max(a+b,a+c)}{2}$

Faith $= \frac{a+0.5d}{a+b+c+d}$

Forbes 1 $= \frac{na}{(a+b)(a+c)}$

Forbes 2 $= \frac{na-(a+b)(a+c)}{n(\min(a+b,a+c))-(a+b)(a+c)}$

Fossum $= \frac{n(a-0.5)^2}{(a+b)(a+c)}$

GilbertWells $= \log(a) - \log(n) - \log(\frac{a+b}{n}) - \log(\frac{a+c}{n})$

GoodmanKruskal $= \frac{\sigma-\sigma'}{2n-\sigma'}$

Gower $= \frac{a+d}{\sqrt{(a+b)(a+c)(b+d)(c+d)}}$

GowerLegendre $= \frac{a+d}{a+0.5(b+c)+d}$

Hamann $= \frac{(a+d)-(b+c)}{a+b+c+d}$

InnerProduct $= a+d$

Intersection $= a$

Jaccard $= \frac{a}{a+b+c}$

3w Jaccard $= \frac{3a}{3a+b+c}$

Johnson $= \frac{a}{a+b}+\frac{a}{a+c}$

Kulczynski 1 $= \frac{a}{(b+c)}$

Kulczynski 2 $= \frac{\frac{a}{2}(2a+b+c)}{(a+b)(a+c)}$

Mcconnaughey $= \frac{a^2-bc}{(a+b)(a+c)}$

Michael $= \frac{4(ad-bc)}{(a+d)^2-(b+c)^2}$

Mountford $= \frac{a}{0.5(ab)+(ac)+(bc)}$

NeiLi $= \frac{2a}{(a+b)(a+c)}$

Ochiai 1 $= \frac{a}{\sqrt{(a+b)(a+c)}}$

Ochiai 2 $= \frac{ad}{\sqrt{(a+b)(a+c)(d+b)(d+c)}}$

PearsonHeron 1 $= \frac{ad-bc}{\sqrt{(a+b)(a+c)(d+b)(d+c)}}$

Pearson 1 $= \chi^2$

Pearson 3 $= \left(\frac{\rho}{n+\rho}\right)^{1/2}$

RogerTanimoto $= \frac{a+d}{a+2(b+c)+d}$

Simpson $= \frac{a}{\min(a+b,a+c)}$

SokalSneath 1 $= \frac{a}{a+2b+2c}$

SokalSneath 3 $= \frac{a+d}{d+c}$

SokalSneath 5 $= \frac{ad}{((a+b)(a+c)(b+d)(c+d))^{0.5}}$

Stiles $= \log_{10} \frac{n(|ad-bc|-\frac{n}{2})^2}{(a+b)(a+c)(b+d)(c+d)}$

Tarantula $= \frac{a(a+b)}{c(c+d)}$

Var of Correlation $= \frac{\log(1+ad)-\log(1+bc)}{\log(1+\frac{n^2}{4})}$

Yulew $= \frac{\sqrt{ad}-\sqrt{bc}}{\sqrt{ad}+\sqrt{bc}}$

Otsuka $= \frac{a}{((a+b)(a+c))^{0.5}}$

PearsonHeron 2 $= cos\left(\frac{\pi\sqrt{bc}}{\sqrt{ad}+\sqrt{bc}}\right)$

Pearson 2 $= \left(\frac{\chi^2}{n+\chi^2}\right)^{1/2}$

Peirce $= \frac{ab+bc}{ab+2bc+cd}$

RusselRao $= \frac{a}{a+b+c+d}$

SokalMichener $= \frac{a+d}{a+b+c+d}$

SokalSneath 2 $= \frac{2(a+d)}{2a+b+c+2d}$

SokalSneath 4 $= \frac{\frac{a}{a+b}+\frac{a}{a+c}+\frac{a}{b+d}+\frac{a}{c+d}}{4}$

Sorgenfrei $= \frac{a^2}{(a+b)(a+c)}$

Tanimoto $= \frac{a}{(a+b)+(a+c)-a}$

Tarwid $= \frac{na-(a+b)(a+c)}{na+(a+b)(a+c)}$

Yuleq $= \frac{ad-bc}{ad+bc}$

where:

$\sigma = \max(a,b) + \max(c,b) + \max(b,d)$

$\sigma' = \max(a+c,b+d) + \max(a+b,c+d)$

$\chi^2 = \frac{n(ad-bc)^2}{(a+b)(a+c)(d+b)(d+c)}$

$\rho = \frac{ad-bc}{\sqrt{(a+b)(a+c)(d+b)(d+c)}}$

APPENDIX B – SSB ON CO-AUTHORSHIPING PREDICTION

As a side experiment of SSB, we will assess here the performance of the SSB measure, which we designed on protein function annotation task based on their 3D structure, in the database described in subsection 4.3.2. One can notice that review features are binary and sparse, similar to protein domains.

As in the protein experiment, we calculate the AP for all models and the result is shown in Table 6. Once again SSB was able to overcome Jaccard. The paired t-test rejects the hypothesis that SSB is statistically the same as Jaccard, whereas it confirms that SSB does not differ from the best similarity.

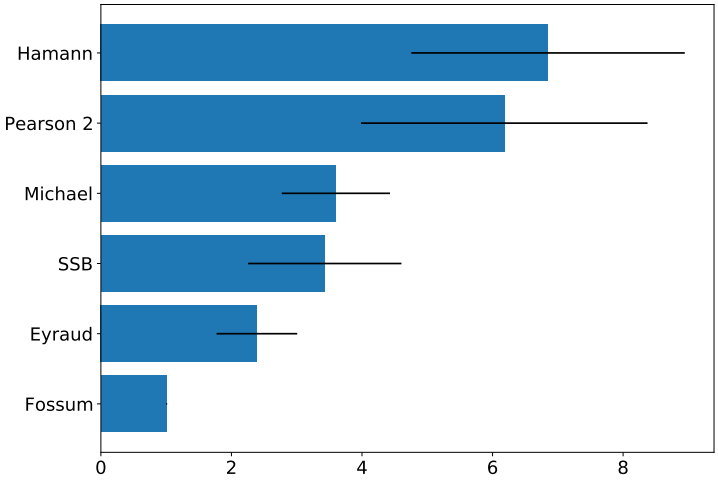
Table 6 – Performance comparison of SSB, Jaccard and the best BSM (Pearson 2) for review's author prediction experiment

	Review's Author Prediction
SSB	0.1724 ± 0.0203
JACCARD	0.1439 ± 0.0176 ✗
BEST	$\mathbf{0.1842 \pm 0.0276}$ ✓

The symbols ✓ and ✗ indicate the result of the hypothesis test (✓ fail to reject, and ✗ reject). The best result is indicated in bold.

We also conducted a test using the random forest algorithm in a similar way that was used in subsection 3.4.1.1. The random forest designed to predict whether two reviews are from the same author reached an average AP of 0.213 ± 0.048 and outperformed all similarities. The feature importance of such random forest is shown in figure 10.

Figure 10 – Feature importance for review's author prediction experiment.



Feature importance for the six best similarities for review's author prediction experiment. Highest values indicate most important features.

Once again SSB was selected as one of the best features for the random forest model. Such fact indicates that SSB has useful information for the prediction task under analysis.

APPENDIX C – PYTHON SIMILARITY LIBRARY

As a sub-product of this study, it was created a python 3 library designed to calculate several similarity functions. The library can use binary vectors or sets as input to calculate the result of 63 BSMs or to return the associated values. Additionally, it provided a practical way of inputting new similarity measures.

Library Source and license - The library can be found in the public repository: <https://github.com/marcelobveras/similarity_measures> under the GNU General Public License v3.0

How to use and examples - After downloading the project folder in the root path of your project import using the code provided in Source Code 1.

Source Code 1 – Importing the library

```
1 import binary_similarity.Similarity_function as sim
```

In Source Code 2, is created an example set and used it as input to create a similarity object. The similarity object contains the associated quantities described in section 2.1.1.

Source Code 2 – Creating Similarity Object

```
1 x = {'elem1', 'elem2', 'elem3'}
2 y = {'elem1', 'elem2', 'elem4', 'elem5'}
3 z = {'elem4', 'elem6'}
4 u = set.union(x, y, z)
5
6 sim_obj = sim.similarity_score(x, y, U=u)
```

To calculate the similarity value, one can specify the function by its name or can calculate them all at once. The first line of Source Code 3 produces a list of all available functions, while the two subsequent lines evaluate the Jaccard function and Dice 1, and the last line creates a dictionary with all the keys being the functions and their values being the evaluation of each function.

Source Code 3 – Evaluating similarities

```
print(sim_obj.listSimFunctions())

print(sim_obj.evaluate('Jaccard'))

print(sim_obj.evaluate('Dice_1'))

print(sim_obj.evaluateAll())

# Output:
# ['Ample', 'Anderberg', 'Baroni_UrbaniBuser_1', '
   Baroni_UrbaniBuser_2', 'BraunBanquet', 'Cosine', 'Dennis
   ', 'Derived_Jaccard', 'Derived_RusellRao', '
   Derived_SokalMichener', 'Derived_logSokalMichener', '
   Dice_1', 'Dice_2', 'Disperson', 'DriverKroeber', 'Eyraud
   ', 'FagerMcGowan', 'Faith', 'Forbes_1', 'Forbes_2', '
   Fossum', 'GilbertWells', 'GoodmanKruskal', 'Gower', '
   GowerLegendre', 'Hamann', 'InnerProduct', 'Intersection
   ', 'Jaccard', 'Jaccard_3w', 'Johnson', 'Kulczynski_1', '
   Kulczynski_2', 'Mcconnaughey', 'Michael', 'Mountford', '
   NeiLi', 'Ochiai_1', 'Ochiai_2', 'Otsuka', '
   PearsonHeron_1', 'PearsonHeron_2', 'Pearson_1', '
   Pearson_2', 'Pearson_3', 'Peirce', 'RogerTanimoto', '
   RusselRao', 'Simpson', 'SokalMichener', 'SokalSneath_1',
    'SokalSneath_2', 'SokalSneath_3', 'SokalSneath_4', '
   SokalSneath_5', 'Sorgenfiel', 'Stiles', 'Tanimoto', '
   Tarantula', 'Tarwid', 'Var_of_Correlation', 'Yuleq', '
   Yulew']

# 0.4

# 0.5714285714285714
```

```
15
16 # {'Ample': 1.0, 'Anderberg': 0.0, 'Baroni_UrbaniBuser_1':
   0.5322887255269254, 'Baroni_UrbaniBuser_2':
   0.06457745105385086, 'BraunBanquet': 0.5, 'Cosine':
   0.5773502691896258, 'Dennis': 0.0, 'Derived_Jaccard':
   0.6131471927654585, 'Derived_RusellRao':
   0.5645750340535797, 'Derived_SokalMichener':
   0.7124143742160444, 'Derived_logSokalMichener':
   0.28758562578395563, 'Dice_1': 0.5714285714285714, '
   Dice_2': 0.2857142857142857, 'Disperson': 0.0, '
   DriverKroeber': 0.5833333333333333, 'Eyraud': 0.0, '
   FagerMcGowan': -1.4226497308103743, 'Faith':
   0.4166666666666667, 'Forbes_1': 16.0, 'Forbes_2': 0.0, '
   Fossum': 18.0, 'GilbertWells': 1.6653345369377348e-16, '
   GoodmanKruskal': 0.0, 'Gower': 0.3535533905932738, '
   GowerLegendre': 0.6666666666666666, 'Hamann': 0.0, '
   InnerProduct': 3.0, 'Intersection': 2.0, 'Jaccard': 0.4,
    'Jaccard_3w': 0.6666666666666666, 'Johnson':
   1.1666666666666665, 'Kulczynski_1': 0.6666666666666666,
   'Kulczynski_2': 0.5833333333333334, 'Mcconnaughey':
   0.16666666666666666, 'Michael': 0.0, 'Mountford': 0.4, '
   NeiLi': 0.5714285714285714, 'Ochiai_1':
   0.5773502691896258, 'Ochiai_2': 0.23570226039551587, '
   Otsuka': 0.5773502691896258, 'PearsonHeron_1': 0.0, '
   PearsonHeron_2': 6.123233995736766e-17, 'Pearson_1':
   0.0, 'Pearson_2': 0.0, 'Pearson_3': 0.0, 'Peirce': 0.5,
   'RogerTanimoto': 0.3333333333333333, 'RusselRao':
   0.3333333333333333, 'Simpson': 0.6666666666666666, '
   SokalMichener': 0.5, 'SokalSneath_1': 0.25, '
   SokalSneath_2': 0.6666666666666666, 'SokalSneath_3':
   1.0, 'SokalSneath_4': 0.49999999999999994, '
   SokalSneath_5': 0.23570226039551587, 'Sorgenfiel':
```

```
   5.333333333333333, 'Stiles': 0.024061024441985674, '
   Tanimoto': 0.4, 'Tarantula': 1.0, 'Tarwid': 0.0, '
   Var_of_Correlation': 0.0, 'Yuleq': 0.0, 'Yulew': 0.0}
```

Creating custom functions is possible as shown in Source Code 4. After defining a new function, using the binary associated values, it can be injected into the library using the method: *setNewFunction*. The injected function will appear as same as other BSMs previously programmed in the source code.

Source Code 4 – Inject other BSMs

```
1  def outsideSimFunc(self):
2      return 2*self.a/(2*self.a+self.b+self.c)
3
4  sim_obj.setNewFunction(outsideSimFunc)
5
6  print(sim_obj.evaluate('outsideSimFunc'))
7
8  print(sim_obj.evaluateAll())
9
10 # Output:
11
12 # 0.5714285714285714
13
14 # {'Ample': 0.75, 'Anderberg': 0.0, 'Baroni_UrbaniBuser_1':
       0.5714285714285714, 'Baroni_UrbaniBuser_2':
   0.14285714285714285, 'BraunBanquet': 0.5, 'Cosine':
   0.5773502691896258, 'Dennis': 0.2182178902359924, '
   Derived_Jaccard': 0.6131471927654585, 'Derived_RusellRao
   ': 0.5283208335737188, 'Derived_SokalMichener':
   0.7739760316291208, 'Derived_logSokalMichener':
   0.3333333333333333, 'Dice_1': 0.5714285714285714, '
   Dice_2': 0.2857142857142857, 'Disperson':
```

```
0.05555555555555555, 'DriverKroeber':
0.5833333333333333, 'Eyraud': 0.6805555555555556, '
FagerMcGowan': -1.4226497308103743, 'Faith':
0.42857142857142855, 'Forbes_1': 18.666666666666668, '
Forbes_2': 0.2222222222222222, 'Fossum': 21.0, '
GilbertWells': 0.15415067982725839, 'GoodmanKruskal':
0.0, 'Gower': 0.3333333333333333, 'GowerLegendre':
0.7272727272727273, 'Hamann': 0.16666666666666666, '
InnerProduct': 4.0, 'Intersection': 2.0, 'Jaccard': 0.4,
 'Jaccard_3w': 0.6666666666666666, 'Johnson':
1.1666666666666665, 'Kulczynski_1': 0.6666666666666666,
'Kulczynski_2': 0.5833333333333334, 'Mcconnaughey':
0.16666666666666666, 'Michael': 0.32, 'Mountford': 0.4,
'NeiLi': 0.5714285714285714, 'Ochiai_1':
0.5773502691896258, 'Ochiai_2': 0.3333333333333333, '
Otsuka': 0.5773502691896258, 'PearsonHeron_1':
0.16666666666666666, 'PearsonHeron_2':
0.2662553420414155, 'Pearson_1': 0.19444444444444445, '
Pearson_2': 0.07329541567217676, 'Pearson_3':
0.15249857033260467, 'Peirce': 0.4, 'RogerTanimoto':
0.4, 'RusselRao': 0.2857142857142857, 'Simpson':
0.6666666666666666, 'SokalMichener': 0.5714285714285714,
 'SokalSneath_1': 0.25, 'SokalSneath_2':
0.7272727272727273, 'SokalSneath_3': 1.3333333333333333,
 'SokalSneath_4': 0.5833333333333333, 'SokalSneath_5':
0.3333333333333333, 'Sorgenfiel': 5.333333333333333, '
Stiles': 0.008314448320316801, 'Tanimoto': 0.4, '
Tarantula': 0.75, 'Tarwid': 0.07692307692307693, '
Var_of_Correlation': 0.20224881438357958, 'Yuleq':
0.3333333333333333, 'Yulew': 0.17157287525380988, '
outsideSimFunc': 0.5714285714285714}
```

Appendix A contains a list of all the BSMs included in the library.