International Journal of
Molecular Sciences

Article

UltraPse: A Universal and Extensible Software Platform for Representing Biological Sequences

Pu-Feng Du [1], Wei Zhao [1], Yang-Yang Miao [1,2], Le-Yi Wei [1] and Likun Wang [3,*]

[1] School of Computer Science and Technology, Tianjin University, Tianjin 300350, China; PufengDu@gmail.com (P.-F.D.); wzhao_cstju@yeah.net (W.Z.); miaoyangyang1998@163.com (Y.-Y.M.); weileyi@tju.edu.cn (L.-Y.W.)
[2] School of Chemical Engineering, Tianjin University, Tianjin 300350, China
[3] Institute of Systems Biomedicine, Beijing Key Laboratory of Tumor Systems Biology, Department of Pathology, School of Basic Medical Sciences, Peking University Health Science Center, Beijing 100191, China
[*] Correspondence: wanglk@hsc.pku.edu.cn; Tel.: +86-10-8280-5807

Received: 10 October 2017; Accepted: 3 November 2017; Published: 14 November 2017

Abstract: With the avalanche of biological sequences in public databases, one of the most challenging problems in computational biology is to predict their biological functions and cellular attributes. Most of the existing prediction algorithms can only handle fixed-length numerical vectors. Therefore, it is important to be able to represent biological sequences with various lengths using fixed-length numerical vectors. Although several algorithms, as well as software implementations, have been developed to address this problem, these existing programs can only provide a fixed number of representation modes. Every time a new sequence representation mode is developed, a new program will be needed. In this paper, we propose the UltraPse as a universal software platform for this problem. The function of the UltraPse is not only to generate various existing sequence representation modes, but also to simplify all future programming works in developing novel representation modes. The extensibility of UltraPse is particularly enhanced. It allows the users to define their own representation mode, their own physicochemical properties, or even their own types of biological sequences. Moreover, UltraPse is also the fastest software of its kind. The source code package, as well as the executables for both Linux and Windows platforms, can be downloaded from the GitHub repository.

Keywords: pseudo-amino acid compositions; pseudo-k nucleotide compositions; extensible software

1. Introduction

Over the last two decades, huge numbers of biological sequences have been deposited in public databases. Until today, the number of these sequences is still increasing exponentially. However, the cellular and functional attributes of these sequences, no matter whether they are nucleotide sequences or protein sequences, remain largely unknown. It is a very important task for computational biology to predict the functional and cellular attributes of these sequences.

In the view of machine learning, most of these prediction tasks can be formulated as pattern classification problems. As elaborated in a series of publications [1–8], one of the most challenging parts is to represent a biological sequence with a fixed-length numerical vector, yet still keep a considerable amount of the sequence-order information. This is because almost every existing algorithm for these tasks can only handle fixed-length vectors, but not the sequences.

For protein and peptide sequences, Chou proposed pseudo-amino acid compositions (PseAAC) [9] and amphiphilic pseudo-amino acid compositions (AmPseAAC) [10]. Ever since the concepts of pseudo-factors were introduced, they have rapidly penetrated into almost every area of computational proteomics [11–20]. As elaborated in a review article, the form of classic pseudo-amino acid compositions has been generalized to contain various types of information [21], which is known as the general-form pseudo-amino acid compositions. The applications of PseAAC concepts have been summarized in the review papers [22,23].

Recently, the concept of PseAAC has been extended to represent nucleotide sequences [24]. Chen et al. developed pseudo-dinucleotide compositions (PseDNC) to predict DNA recombination hostspots [25]. This formulation was then extended as pseudo-k nucleotide compositions (PseKNC), which have been applied in predicting splicing sites [26], predicting translation initiation sites [27], predicting nucleosome positions [28], predicting promoters [29], predicting DNA methylation sites [30], predicting microRNA precursors [31] and many others [32–41].

In the early days of pseudo-amino acid compositions, every study had to implement PseAAC independently. Although the algorithms in every implementation are identical, different implementations may introduce computational discrepancies due to technical details. For example, different implementations may give results with different precisions. This kind of differences may be amplified by machine-learning based predictors, which may eventually produce different prediction results. For another example, different implementations may have very different computational efficiencies. This means one implementation may only use a second to process a dataset, while another program may require over an hour to achieve the same results on the same dataset with the same parameters.

To solve these problems, a universal implementation of the algorithm should be provided. Many efforts have been made for this purpose [42–52]. The first program focus on the PseAAC formulation is the PseAAC server [43], which was brought online in the year 2008. The PseAAC server can compute Type-I and Type-II PseAAC using six different kinds of physicochemical properties of amino acids. The PseAAC server has a friendly user interface, which is convenient and efficient for small datasets. However, for large datasets and the repeatedly parameter scanning process, the computational efficiency of the PseAAC server is not ideal. The PseAAC-Builder [45], which was released in the year 2012, is dedicated to improving the efficiency. Unlike the PseAAC server, the PseAAC-Builder is a stand-alone program that can be executed locally. It has a simple graphical user interface (GUI) for the users' convenience. It can also be executed in a command line environment. The computational efficiency of PseAAC-Builder is much higher than the PseAAC server, especially in the command line environment. Although the PseAAC-Builder includes over 500 different types of physicochemical properties, it did not provide the ability to compute general form PseAAC. PseAAC-General [46], which is a major upgrade to the PseAAC-Builder, was developed to solve this problem. PseAAC-General provides the ability to compute several commonly used general forms of PseAAC, such as the GO mode, the functional domain mode and the evolutionary mode. The users of PseAAC-General can slightly extend its ability by using Lua scripts.

After Chen et al. proposed the PseKNC representations for nucleotide sequences, similar software and services were needed for DNA and RNA sequences. Chen et al. released the PseKNC [48] and PseKNC-General [49] packages for converting DNA/RNA sequences into its PseKNC or general form of PseKNC representations. Liu et al. developed the repDNA [50], repRNA [51], and Pse-In-One [52] services for more types of descriptors. The Pse-In-One service attempts to be a universal online service that can be applied on both protein and nucleotide sequences.

However, all existing software packages and online services suffer from three problems. (1) Lack of extensibility. Most of the existing software can only be used to produce existing modes of representation. The users cannot extend the software to handle their own novel representation modes. Although PseAAC-General can be extended by using Lua script, it can only be used for protein sequences; (2) Lack of flexibility. Most of the existing software can only handle one type of biological sequences, either nucleotide sequences or protein sequences. Pse-In-One is the only existing service that can handle protein sequence as well as nucleotide sequences. However, no program can handle user-defined sequence types. For example, when studying the protein phosphorylation sites, the modified residues should have different notations of sequences, which are not in the standard 20 letters. The users need to define the extra letters to represent the modified residues. As far as we know, no program can handle this kind of sequence; (3) Lack of computational efficiency on large datasets. Most of the existing programs are not designed to handle large datasets. They may need many minutes to process a million sequences. If a user needs to repeatedly scan parameters of a representation, the processing time may be days or even weeks.

In this paper, we proposed the UltraPse program, which is a universal and extensible software platform for all possible sequence representation modes. The UltraPse program unified the processing of nucleotide and protein sequence in one program, as well as the user-defined sequence types. UltraPse supports two forms of extension modules, the BSOs (Binary Shared Objects) and the Lua scripts, which are called the TDFs (Task Definition Files) in UltraPse. The users can develop their own modes by just writing several lines of Lua scripts. UltraPse has very high computational efficiency. It is even faster than the PseAAC-General, which used to be the fastest program of its kind. For the users' convenience, we have integrated many existing modes within the UltraPse. We expect that the UltraPse program can be a useful platform which simplifies all future programming works in developing novel sequence representation modes. All source codes of UltraPse, including some extension modules can be downloaded freely under the term of GNU GPL (GNU General Public License) v3 from the GitHub repository: https://github.com/pufengdu/UltraPse.

2. Results and Discussion

2.1. Computational Efficiency Analysis

We compared the computational efficiency of UltraPse to that of PseAAC-General and Pse-In-One under the same conditions. As in Figure 1, the UltraPse can process over 120 thousand sequences per second, while PseAAC-General can process about 85 thousand sequences per second. Unfortunately, the Pse-In-One can process only less than one thousand sequences per second. According to these results, the computational efficiency of UltraPse is roughly 1.5 times of the PseAAC-General, and about 185 times of Pse-In-One. Since the algorithms of the three programs are essentially the same, the reason for the efficiency differences resides in the technical details of the implementations.

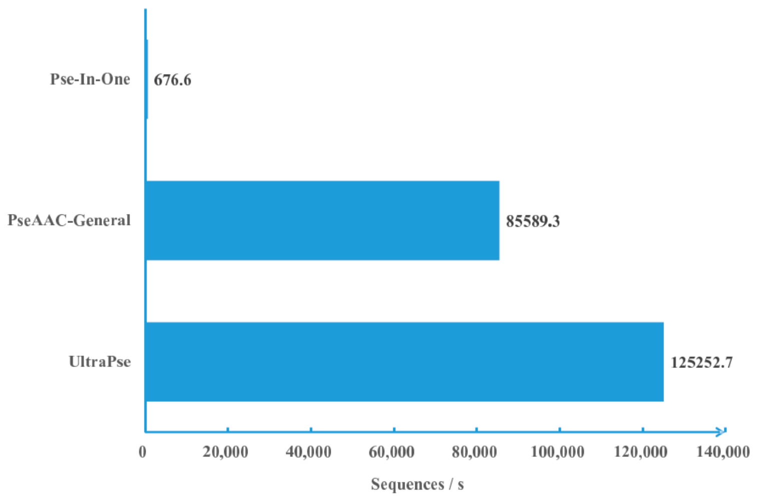


Figure 1. Computational efficiency comparisons. Three programs are compared. The comparison was carried out by letting the three programs compute amino acid compositions on the same dataset on the same machine. Every program was executed with the same parameters for three times. The average execution time was applied in calculating the computational efficiency. The computational efficiency is measured by the average number of sequences that are processed every second. Pse-In-One: A program in literature [52]; PseAAC-General: A program in literature [46]; UltraPse: A program of this work.

2.2. Flexibility and Extensibility

We integrated 35 sequence representation modes within the UltraPse. The representation modes can be organized hierarchically as in Figure 2. The integrated modes can be used to represent protein, as well as DNA and RNA sequences. The modes cover most of the representation modes that can be generated by PseAAC-General, PseKNC-General, and Pse-In-One. Moreover, UltraPse can generate even more modes, for example, the commonly used one-hot encoding mode [53–55]. The sequence representation modes of UltraPse can be extended by using BSOs and TDFs. According to our own works, using UltraPse in developing novel representation modes can save over half of the programing labor.

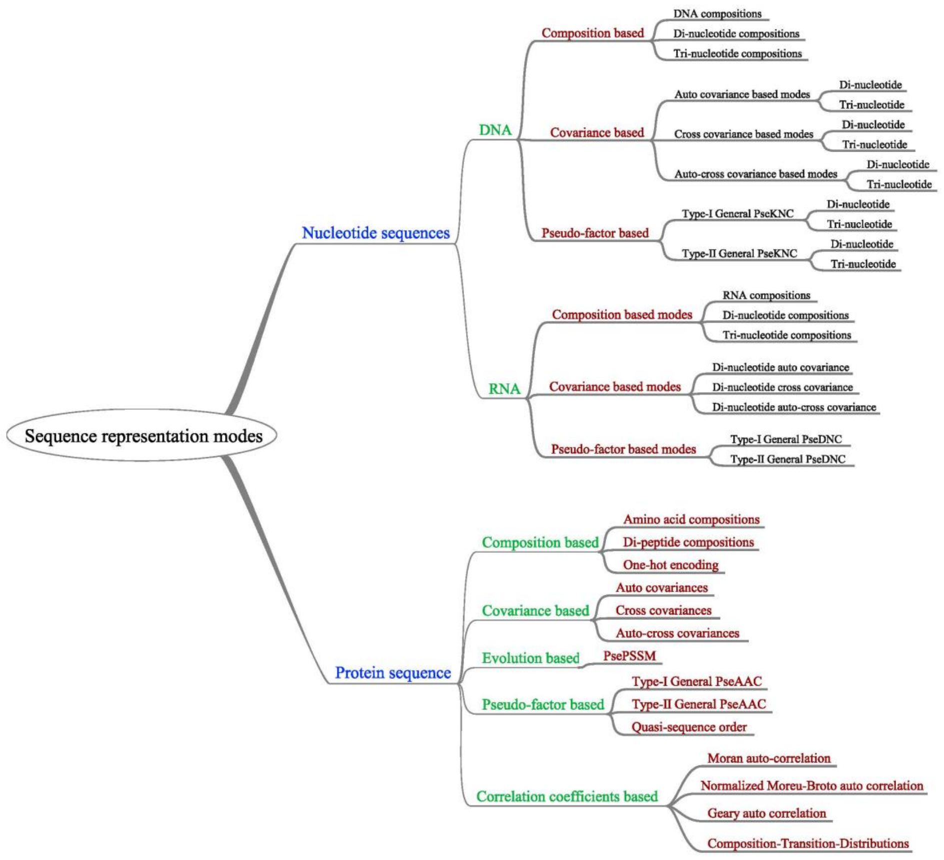


Figure 2. Hierarchical organization of integrated sequence representation modes. UltraPse integrated the sequence representation modes in its distribution package. Most of these modes can also be applied in user-defined sequence types, as long as the users provide proper definitions of the physicochemical properties.

Besides the user-defined representation modes of protein and nucleotide sequences, the users of UltraPse can define their own sequence types using TDFs. They are allowed to choose a set of letters other than the standard ones to represent additional information. For example, a user can use C for cytidines on a DNA sequence, and M for methylated cytidines. The choice of the letter M totally depends on the users. Even more, the users of UltraPse can define their own physicochemical properties with TDFs.

The TDFs of UltraPse is written using Lua language, which is a simple, powerful and extensible programming language which has been applied in bioinformatics software previously [56]. We provide over 20 UltraPse specific functions and interfaces. Users can access and modify UltraPse internal data structures using these functions in TDFs. We compared the flexibility and the extensibility of different software in Table 1.

Table 1. Software function comparison in terms of flexibility and extensibility.

Software Functions	Sequence Types	Extensibility
UltraPse	DNA, RNA, Protein, User-defined types	Users can define their own sequence types, representation modes and physicochemical properties
PseAAC-General [46]	Protein	Users can define their own representation modes
PseAAC-Builder [45]	Protein	No extensibility
Pse-In-One [52]	DNA, RNA, Protein	Users can define their own physicochemical properties
PseKNC [48]	DNA, RNA	Users can define their own physicochemical properties
PseKNC-General [49]	DNA, RNA	Users can define their own physicochemical properties

2.3. Compatibility and Robustness

UltraPse can recognize FASTA format files that are directly downloaded from one of the following five databases: GenBank, UniProt, EMBL, DDBJ, and RefSeq. The sequence identifiers and comments in these public databases can be automatically recognized. For FASTA file that are not from these public databases, UltraPse can also recognize them as long as the comment line of every sequence is unique in the FASTA file. Besides the FASTA format requirements, there is no additional restriction on input data format. As indicated in Table 2, this is a unique advantage of UltraPse.

According to Chou's five step rule [12,21,57–60], before converting biological sequences into numerical vectors, a high-quality benchmark dataset must be constructed. The construction of a dataset usually includes a step to filter out the sequences containing non-standard letters. For example, B, J, or X appear in protein sequences in the UniProt database. However, the sequences containing these letters are hardly suitable for further analysis in many cases. As indicated in Table 2, UltraPse provides a user-controllable data fault tolerant ability. According to users' choice, when one of these sequences is encountered, UltraPse can automatically skip the sequence or abort all further computations. This function is useful in adopting third-party datasets in practical works, because filtering out the sequences usually requires tedious programming work.

Table 2. Software function comparison in terms of data processing ability.

Software	Output Formats	Input Formats	Data Fault Tolerant [a]
UltraPse	SVM [b], TSV [c], CSV [d]	Multi-line FASTA (Automatic ID recognition for UniProt, GenBank, EMBL, DDBJ and RefSeq)	User-controllable behavior on data faults
PseAAC-General [46]	SVM, TSV, CSV	Single-line FASTA (With restrictions on comment line) [e]	Automatically ignore and report data faults
PseAAC-Builder [45]	SVM, TSV, CSV	Single-line FASTA (With restrictions on comment line)	Automatically ignore and report data faults
Pse-In-One [52]	SVM, TSV, CSV	Mutlti-line FASTA	Abort processing on data faults
PseKNC [48]	SVM, TSV, CSV	Mutlti-line FASTA	Abort processing on data faults
PseKNC-General [49]	SVM, TSV, CSV	Mutlti-line FASTA	Abort processing on data faults

[a] Data fault tolerant: The behavior of a software when it encounters some invalid data records. Here, the invalid data records include the sequences with non-standard letter and the sequence without sufficient length; [b] SVM: data format for libSVM [61]; [c] TSV: tab separated vector; [d] CSV: comma separated vector; [e] Single-line FASTA: the sequence of a record in the file must not spread to multiple lines. Both PseAAC-General and PseAAC-Builder have the same restrictions.

2.4. Technical Detail Comparison

Most state-of-the art software is written in Python, while PseAAC-General and UltraPse are written in C++. This difference eventually made the difference in computational efficiency. Since the computational efficiencies of PseAAC-General and UltraPse are comparable, we can compare several technical details of them.

PseAAC-General is a program that can be extended by using Binary Extension Modules (BEMs). However, it should be noted that, the BEMs of PseAAC-General are completely different to the BSOs in UltraPse. A BEM of PseAAC-General is just a compressed data block. However, how this data block should be used, was still implemented by the PseAAC-General main program. In the UltraPse, a BSO is actually a dynamically loaded library, which contains all the information and instructions for constructing one or more sequence representation modes. Therefore, the BSOs of UltraPse are much more flexible than the BEMs of PseAAC-General.

We have seen that UltraPse has roughly 1.5 times the efficiency of PseAAC-General. This advantage is achieved by an internal representation scheme and a pre-computing mechanism of UltraPse. In PseAAC-General, the sequences are converted to a series of physicochemical properties. The sequence descriptors are then computed according to the corresponding algorithms. However, this intuitive implementation requires repeatedly computing dot-product or Euclidean distance between physicochemical vectors of different amino acids. Since the combination of two different amino acids is limited, we pre-compute the dot-product and Euclidean distance for all possible combinations in UltraPse. The sequences in UltraPse are not converted into a series of physicochemical properties. They are converted into UltraPse internal indices, which can be used to quickly find correct values that have been pre-computed. When computing only the amino acids compositions, the implementations of PseAAC-General and UltraPse are similar. However, UltraPse still benefits from converting all sequences into internal indices first. Because, the amino acids counting procedure becomes simpler, this allows the compiler to do more optimization for speed. This is why UltraPse is faster than PseAAC-General.

2.5. Future Works in Plan

Besides the practical application of UltraPse program in research projects, there is still much work to do in terms of software development. The work at first priority is to add an automated unit-testing facility in the source code of UltraPse. Unit-testing is good practice in software engineering to ensure robustness of large scale software. It will be very important for the future versions of UltraPse. The next work in plan is to enable UltraPse support more data formats as input files. As far as we can tell, no existing program in representing biological sequences can handle file formats other than FASTA. We will make the next version of UltraPse handle FASTA, FASTQ, and several other formats of input file.

2.6. Availability

The UltraPse software is provided as source codes and binary packages. All the source codes can be downloaded from the GitHub repository (Available online: https://github.com/pufengdu/UltraPse). The binary distribution packages can also be downloaded from the Release sub-directory in the GitHub repository. Currently, there are binary packages for Windows and Linux platforms. The Windows binary program can be executed directly. The Linux binary package has been tested on a freshly installed Ubuntu Linux Server 16.04.3.

3. Methods

3.1. Efficiency Comparison Protocols

We performed computational efficiency comparisons on a server with an Intel Xeon X3470 processor and 32 GB memory. To perform a fair comparison, we installed Pse-In-One locally on the server. We also locally compiled and installed PseAAC-General and UltraPse on the same server.

The testing dataset is the "huge" testing dataset that can be obtained from the official website of PseAAC-General. This dataset contains 516,081 protein sequences. Since the Pse-In-One keeps complaining about non-standard letters and too short sequences in the dataset, we excluded all the sequences that have non-standard letters. The remaining 513,536 protein sequences were fed into three programs independently. All three programs are configured to compute only amino acid compositions. The computational times are measured by the "real" time value of the standard Linux time command. To eliminate random errors, every program was executed consecutively with exactly the same configuration three times. The average computational time was used in calculating computational efficiency.

3.2. Abstracted Software Design

We illustrate the internal structure and the data-flows of UltraPse in Figure 3. There are four major parts within UltraPse. They are the FASTA parser, sequence preprocessor, computing engine, and the result writer. The FASTA parser is responsible for loading FASTA format sequences into the memory from a hard drive. It also organizes the sequences according to their identifiers and their sequence types. These sequences are then sent to the sequence preprocessor, where the sequences are converted to UltraPse internal indices according to the sequence type definitions. The computing engine is composed of several mode modules, which are configured according to user requirements. The internal indices go through all mode modules. Eventually, sequence descriptors are generated. The result writer exports these descriptors on the hard drive according to the format requirements.

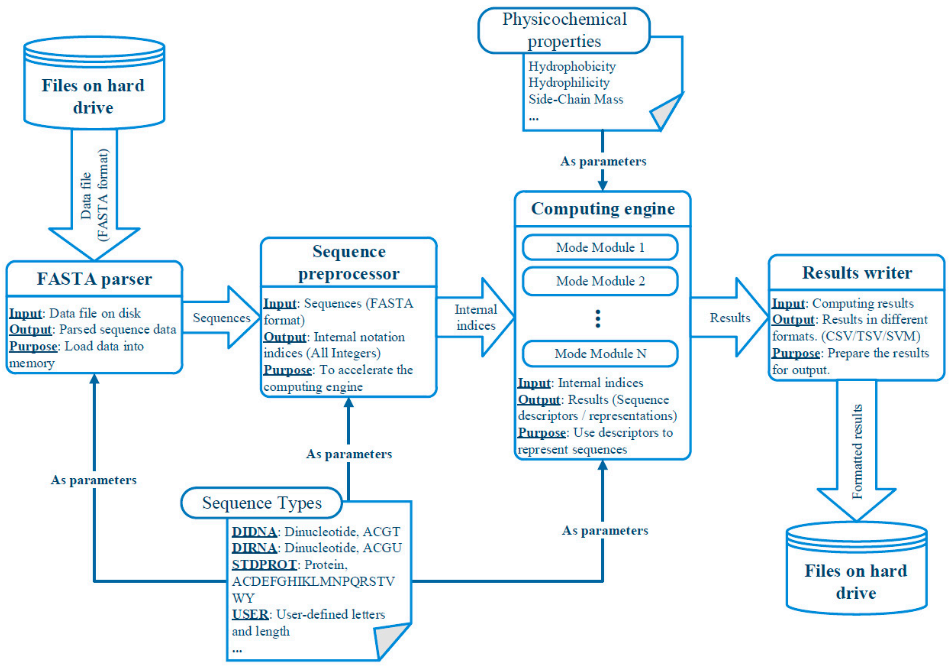

Files on hard drive
Data file (FASTA format)
FASTA parser
Input: Data file on disk
Output: Parsed sequence data
Purpose: Load data into memory
Sequences
Sequence preprocessor
Input: Sequences (FASTA format)
Output: Internal notation indices (All Integers)
Purpose: To accelerate the computing engine
Internal indices
Physicochemical properties
Hydrophobicity
Hydrophilicity
Side-Chain Mass
...
As parameters
Computing engine
Mode Module 1
Mode Module 2
Mode Module N
Input: Internal indices
Output: Results (Sequence descriptors / representations)
Purpose: Use descriptors to represent sequences
Results
Results writer
Input: Computing results
Output: Results in different formats. (CSV/TSV/SVM)
Purpose: Prepare the results for output.
Formatted results
Files on hard drive
As parameters
As parameters
As parameters
Sequence Types
DIDNA: Dinucleotide, ACGT
DIRNA: Dinucleotide, ACGU
STDPROT: Protein, ACDEFGHIKLMNPQRSTVWY
USER: User-defined letters and length
...

Figure 3. The abstracted software design and data flow chart of UltraPse.

3.3. Implementation Technology

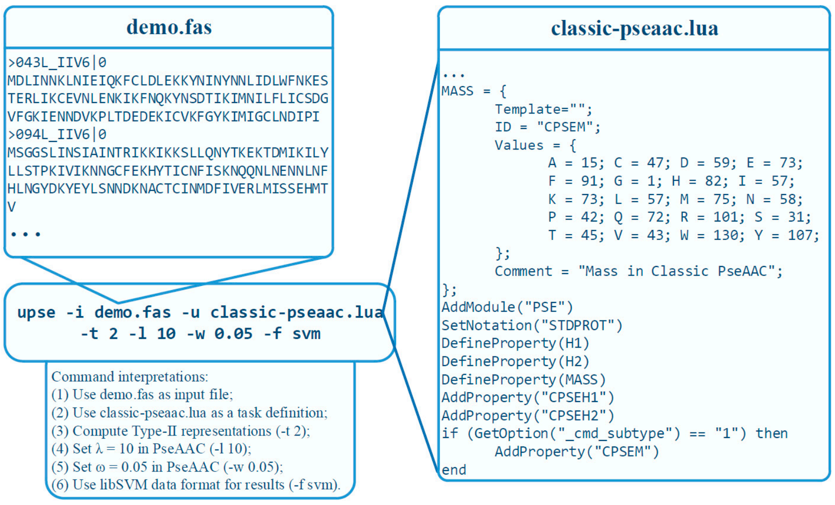
The UltraPse main program is written using standard C++ language, following C++14 standard. The destination hardware architecture is x86-64. The dependencies of the UltraPse main program include GNU standard C library and the embedded interpreter of Lua scripting language. The BSOs of UltraPse are also written using C++, following the same rules as the main program.

On the Linux platform, the compiler for producing binary executables is the GNU g++ version 7.2. The users should first install Lua scripting language. The configuration and compilation of UltraPse need the library provided by the Lua package. On the Windows platform, the MinGW64 version g++ compiler is applied. Several independent libraries are required to compile the codes. For the convenience of Windows users, we provide a binary executable package for the Windows platform.

The TDFs are provided as platform-independent Lua scripts, which can be viewed, edited, and loaded as their original form. The internal data structures of UltraPse can be accessed by Lua scripts using UltraPse specific functions and interfaces. The details on how to write TDFs can be found in the software manual.

3.4. A Practical Example

Figure 4 demonstrate a practical example. The classic pseudo-amino acid composition modes, including type-I and type-II, are implemented using a TDF in UltraPse. The TDF for classic pseudo-amino acid compositions can be found in the "tdfs" subdirectory of UltraPse. The right part of Figure 4 is a part of this TDF. With this TDF, the users only need to specify some parameters on the command line. For example, the "–l 10 –w 0.05" on the command line indicate the value of λ and ω in the PseAAC formulations. Unlike PseAAC-General, where the meanings of all command line options are fixed, the meanings of command line options can be altered by the TDFs in the UltraPse. This is to simplify the development of novel sequence representation modes, where parameters are required to perform correct and efficient computations.



demo.fas

```
>043L_IIV6|0
MDLINNKLNIEIQKFCLDLEKKYNINYNNLIDLWFNKES
TERLIKCEVNLENKIKFNQKYNSDTIKIMNILFLICSDG
VFGKIENNDVKPLTDEDEKICVKFGYKIMIGCLNDIPI
>094L_IIV6|0
MSGGSLINSIAINTRIKKIKKSLLQNYTKEKTDMIKILY
LLSTPKIVIKNNGCFEKHYTICNFISKNQQNLNENNLNF
HLNGYDKYEYLSNNDKNACTCINMDFIVERLMISSEHMT
V

...
```

```
upse -i demo.fas -u classic-pseaac.lua
     -t 2 -l 10 -w 0.05 -f svm
```

Command interpretations:
(1) Use demo.fas as input file;
(2) Use classic-pseaac.lua as a task definition;
(3) Compute Type-II representations (-t 2);
(4) Set λ = 10 in PseAAC (-l 10);
(5) Set ω = 0.05 in PseAAC (-w 0.05);
(6) Use libSVM data format for results (-f svm).

classic-pseaac.lua

```
...
MASS = {
	Template="";
	ID = "CPSEM";
	Values = {
		A = 15; C = 47; D = 59; E = 73;
		F = 91; G = 1; H = 82; I = 57;
		K = 73; L = 57; M = 75; N = 58;
		P = 42; Q = 72; R = 101; S = 31;
		T = 45; V = 43; W = 130; Y = 107;
	};
	Comment = "Mass in Classic PseAAC";
};
AddModule("PSE")
SetNotation("STDPROT")
DefineProperty(H1)
DefineProperty(H2)
DefineProperty(MASS)
AddProperty("CPSEH1")
AddProperty("CPSEH2")
if (GetOption("_cmd_subtype") == "1") then
	AddProperty("CPSEM")
end
```

Figure 4. An example on using UltraPse. UltraPse was used to implement classic pseudo-amino acid compositions. A TDF: classic-pseaac.lua, was applied. The FASTA format sequences are stored in the demo.fas file. The command options indicate that the Type 2 PseAAC will be computed with parameters: λ = 10 and ω = 0.05. The output format is compatible to libSVM.

4. Conclusions

In this paper, we described our new software, the UltraPse (Available online: https://github.com/pufengdu/UltraPse). UltraPse is a universal and extensible software platform for generating biological sequence representations. Since many programs have already been released for various sequence representations, UltraPse has no intention to be a new competitor on the same playground. We expect that UltraPse can work side-by-side with other existing programs, such as PseAAC-General, PseAAC-Builder and Pse-In-One, to accelerate the process of generating sequence representations under various working environments.

Although we have integrated many existing sequence modes within the UltraPse, it should be noted that the major advantage of UltraPse is its flexibility and extensibility. It was designed to be a software platform rather than a program with specific functions. It aims at simplifying all future programming works in developing novel sequence representations.

Web servers have already been proved to be a good method in releasing software. However, presenting UltraPse with a web server will severely damage its computational efficiency. Therefore, we do not provide an online web server for UltraPse. We would rather provide it as a local program. The users need to compile and install it on their own servers. The graphical user interfaces (GUI) is useful on platforms like Microsoft Windows. We will develop a GUI for UltraPse on the Windows platform in future.

Acknowledgments: This work is funded by the National Natural Science Foundation of China (NSFC 31401132); National Natural Science Foundation of China (NSFC 61005041); Tianjin Natural Science Foundation (No. 12JCQNJC02300).

Author Contributions: Pu-Feng Du designed the software, wrote most of the codes and the paper in part. Wei Zhao partially wrote the codes and in part tested the software. Yang-Yang Miao partially tested the software and in part wrote the paper. Le-Yi Wei partially wrote the paper. Likun Wang provided technical discussions, and in part wrote the code and the paper.

Conflicts of Interest: The authors declare no conflict of interest.

Abbreviations

AmPseAAC	Amphiphilic pseudo amino acid composition
BEM	Binary extension module
BSO	Binary shared objects
GPL	General public license
GUI	Graphical user interfaces
PseAAC	Pseudo-amino acid composition
PseDNC	Pseudo-dinucleotide composition
PseKNC	Pseudo-k nucleotide composition
TDF	Task definition file

References

1. Jiao, Y.-S.; Du, P.-F. Predicting Golgi-resident protein types using pseudo amino acid compositions: Approaches with positional specific physicochemical properties. *J. Theor. Biol.* **2016**, *391*, 35–42. [CrossRef] [PubMed]
2. Jiao, Y.-S.; Du, P.-F. Predicting protein submitochondrial locations by incorporating the positional-specific physicochemical properties into Chou's general pseudo-amino acid compositions. *J. Theor. Biol.* **2017**, *416*, 81–87. [CrossRef] [PubMed]
3. Nanni, L.; Brahnam, S.; Lumini, A. High performance set of PseAAC and sequence based descriptors for protein classification. *J. Theor. Biol.* **2010**, *266*, 1–10. [CrossRef] [PubMed]
4. Nanni, L.; Brahnam, S.; Lumini, A. Prediction of protein structure classes by incorporating different protein descriptors into general Chou's pseudo amino acid composition. *J. Theor. Biol.* **2014**, *360*, 109–116. [CrossRef] [PubMed]
5. Li, L.; Yu, S.; Xiao, W.; Li, Y.; Hu, W.; Huang, L.; Zheng, X.; Zhou, S.; Yang, H. Protein submitochondrial localization from integrated sequence representation and SVM-based backward feature extraction. *Mol. Biosyst.* **2014**, *11*, 170–177. [CrossRef] [PubMed]
6. Lin, H.; Chen, W.; Yuan, L.-F.; Li, Z.-Q.; Ding, H. Using Over-Represented Tetrapeptides to Predict Protein Submitochondria Locations. *Acta Biotheor* **2013**, *61*, 259–268. [CrossRef] [PubMed]
7. Zuo, Y.-C.; Peng, Y.; Liu, L.; Chen, W.; Yang, L.; Fan, G.-L. Predicting peroxidase subcellular location by hybridizing different descriptors of Chou' pseudo amino acid patterns. *Anal. Biochem.* **2014**, *458*, 14–19. [CrossRef] [PubMed]
8. Nanni, L.; Lumini, A.; Gupta, D.; Garg, A. Identifying Bacterial Virulent Proteins by Fusing a Set of Classifiers Based on Variants of Chou's Pseudo Amino Acid Composition and on Evolutionary Information. *IEEE-ACM Trans. Comput. Biol. Bioinform.* **2012**, *9*, 467–475. [CrossRef] [PubMed]

9. Chou, K.-C. Prediction of protein cellular attributes using pseudo-amino acid composition. *Proteins* **2001**, *43*, 246–255. [CrossRef] [PubMed]
10. Chou, K.-C. Using amphiphilic pseudo amino acid composition to predict enzyme subfamily classes. *Bioinformatics* **2005**, *21*, 10–19. [CrossRef] [PubMed]
11. Chou, K.-C. Pseudo Amino Acid Composition and its Applications in Bioinformatics, Proteomics and System Biology. *Curr. Proteom.* **2009**, *6*, 262–274. [CrossRef]
12. Qiu, W.-R.; Sun, B.-Q.; Xiao, X.; Xu, Z.-C.; Chou, K.-C. iHyd-PseCp: Identify hydroxyproline and hydroxylysine in proteins by incorporating sequence-coupled effects into general PseAAC. *Oncotarget* **2016**, *7*, 44310–44321. [CrossRef] [PubMed]
13. Xu, Y.; Ding, Y.-X.; Ding, J.; Wu, L.-Y.; Deng, N.-Y. Phogly–PseAAC: Prediction of lysine phosphoglycerylation in proteins incorporating with position-specific propensity. *J. Theor. Biol.* **2015**, *379*, 10–15. [CrossRef] [PubMed]
14. Jia, J.; Zhang, L.; Liu, Z.; Xiao, X.; Chou, K.-C. pSumo-CD: Predicting sumoylation sites in proteins with covariance discriminant algorithm by incorporating sequence-coupled effects into general PseAAC. *Bioinformatics* **2016**, *32*, 3133–3141. [CrossRef] [PubMed]
15. Ahmad, K.; Waris, M.; Hayat, M. Prediction of Protein Submitochondrial Locations by Incorporating Dipeptide Composition into Chou's General Pseudo Amino Acid Composition. *J. Membr. Biol.* **2016**, *249*, 293–304. [CrossRef] [PubMed]
16. Feng, P.-M.; Chen, W.; Lin, H.; Chou, K.-C. iHSP-PseRAAAC: Identifying the heat shock protein families using pseudo reduced amino acid alphabet composition. *Anal. Biochem.* **2013**, *442*, 118–125. [CrossRef] [PubMed]
17. Lin, W.-Z.; Fang, J.-A.; Xiao, X.; Chou, K.-C. iLoc-Animal: A multi-label learning classifier for predicting subcellular localization of animal proteins. *Mol. Biosyst.* **2013**, *9*, 634–644. [CrossRef] [PubMed]
18. Mohabatkar, H.; Mohammad Beigi, M.; Esmaeili, A. Prediction of GABAA receptor proteins using the concept of Chou's pseudo-amino acid composition and support vector machine. *J. Theor. Biol.* **2011**, *281*, 18–23. [CrossRef] [PubMed]
19. Jiao, Y.-S.; Du, P.-F. Prediction of Golgi-resident protein types using general form of Chou's pseudo-amino acid compositions: Approaches with minimal redundancy maximal relevance feature selection. *J. Theor. Biol.* **2016**, *402*, 38–44. [CrossRef] [PubMed]
20. Du, P.; Wang, L. Predicting human protein subcellular locations by the ensemble of multiple predictors via protein-protein interaction network with edge clustering coefficients. *PLoS ONE* **2014**, *9*, e86879. [CrossRef] [PubMed]
21. Chou, K.-C. Some remarks on protein attribute prediction and pseudo amino acid composition. *J. Theor. Biol.* **2011**, *273*, 236–247. [CrossRef] [PubMed]
22. Chou, K.-C. Some remarks on predicting multi-label attributes in molecular biosystems. *Mol. Biosyst.* **2013**, *9*, 1092–1100. [CrossRef] [PubMed]
23. Chou, K.-C. Impacts of bioinformatics to medicinal chemistry. *Med. Chem.* **2015**, *11*, 218–234. [CrossRef] [PubMed]
24. Chen, W.; Lin, H.; Chou, K.-C. Pseudo nucleotide composition or PseKNC: An effective formulation for analyzing genomic sequences. *Mol. Biosyst.* **2015**, *11*, 2620–2634. [CrossRef] [PubMed]
25. Chen, W.; Feng, P.-M.; Lin, H.; Chou, K.-C. iRSpot-PseDNC: Identify recombination spots with pseudo dinucleotide composition. *Nucleic. Acids Res.* **2013**, *41*, e68. [CrossRef] [PubMed]
26. Chen, W.; Feng, P.-M.; Lin, H.; Chou, K.-C. iSS-PseDNC: Identifying splicing sites using pseudo dinucleotide composition. *Biomed. Res. Int.* **2014**, *2014*, 623149. [CrossRef] [PubMed]
27. Chen, W.; Feng, P.-M.; Deng, E.-Z.; Lin, H.; Chou, K.-C. iTIS-PseTNC: A sequence-based predictor for identifying translation initiation site in human genes using pseudo trinucleotide composition. *Anal. Biochem.* **2014**, *462*, 76–83. [CrossRef] [PubMed]
28. Guo, S.-H.; Deng, E.-Z.; Xu, L.-Q.; Ding, H.; Lin, H.; Chen, W.; Chou, K.-C. iNuc-PseKNC: A sequence-based predictor for predicting nucleosome positioning in genomes with pseudo k-tuple nucleotide composition. *Bioinformatics* **2014**, *30*, 1522–1529. [CrossRef] [PubMed]
29. Lin, H.; Deng, E.-Z.; Ding, H.; Chen, W.; Chou, K.-C. iPro54-PseKNC: A sequence-based predictor for identifying sigma-54 promoters in prokaryote with pseudo k-tuple nucleotide composition. *Nucleic Acids Res.* **2014**, *42*, 12961–12972. [CrossRef] [PubMed]

30. Chang, C.-C.; Lin, C.-J.; Chen, W.; Feng, P.; Ding, H.; Lin, H.; Chou, K.-C. iRNA-Methyl: Identifying N^6-methyladenosine sites using pseudo nucleotide composition. *Anal. Biochem.* **2015**, *490*, 26–33. [CrossRef]
31. Liu, B.; Fang, L.; Liu, F.; Wang, X.; Chou, K.-C. iMiRNA-PseDPC: MicroRNA precursor identification with a pseudo distance-pair composition approach. *J. Biomol. Struct. Dyn.* **2016**, *34*, 223–235. [CrossRef] [PubMed]
32. Chen, W.; Tang, H.; Ye, J.; Lin, H.; Chou, K.-C. iRNA-PseU: Identifying RNA pseudouridine sites. *Mol. Ther. Nucleic Acids* **2016**, *5*, e332. [CrossRef] [PubMed]
33. Liu, B.; Long, R.; Chou, K.-C. iDHS-EL: Identifying DNase I hypersensitive sites by fusing three different modes of pseudo nucleotide composition into an ensemble learning framework. *Bioinformatics* **2016**, *32*, 2411–2418. [CrossRef] [PubMed]
34. Liu, B.; Yang, F.; Huang, D.-S.; Chou, K.-C. iPromoter-2L: A two-layer predictor for identifying promoters and their types by multi-window-based PseKNC. *Bioinformatics* **2017**. [CrossRef] [PubMed]
35. Iqbal, M.; Hayat, M. "iSS-Hyb-mRMR": Identification of splicing sites using hybrid space of pseudo trinucleotide and pseudo tetranucleotide composition. *Comput. Methods Programs Biomed.* **2016**, *128*, 1–11. [CrossRef] [PubMed]
36. Kabir, M.; Iqbal, M.; Ahmad, S.; Hayat, M. iTIS-PseKNC: Identification of Translation Initiation Site in human genes using pseudo k-tuple nucleotides composition. *Comput. Biol. Med.* **2015**, *66*, 252–257. [CrossRef] [PubMed]
37. Zhang, M.; Sun, J.-W.; Liu, Z.; Ren, M.-W.; Shen, H.-B.; Yu, D.-J. Improving N(6)-methyladenosine site prediction with heuristic selection of nucleotide physical-chemical properties. *Anal. Biochem.* **2016**, *508*, 104–113. [CrossRef] [PubMed]
38. Dong, C.; Yuan, Y.-Z.; Zhang, F.-Z.; Hua, H.-L.; Ye, Y.-N.; Labena, A.A.; Lin, H.; Chen, W.; Guo, F.-B. Combining pseudo dinucleotide composition with the Z curve method to improve the accuracy of predicting DNA elements: A case study in recombination spots. *Mol. Biosyst.* **2016**, *12*, 2893–2900. [CrossRef] [PubMed]
39. Liu, B.; Liu, Y.; Huang, D. Recombination Hotspot/Coldspot Identification Combining Three Different Pseudocomponents via an Ensemble Learning Approach. *Biomed. Res. Int.* **2016**, *2016*, 8527435. [CrossRef] [PubMed]
40. Qiu, W.-R.; Jiang, S.-Y.; Xu, Z.-C.; Xiao, X.; Chou, K.-C. iRNAm5C-PseDNC: Identifying RNA 5-methylcytosine sites by incorporating physical-chemical properties into pseudo dinucleotide composition. *Oncotarget* **2017**, *8*, 41178–41188. [CrossRef] [PubMed]
41. Xu, Z.-C.; Wang, P.; Qiu, W.-R.; Xiao, X. iSS-PC: Identifying Splicing Sites via Physical-Chemical Properties Using Deep Sparse Auto-Encoder. *Sci. Rep.* **2017**, *7*, 8222. [CrossRef] [PubMed]
42. Li, Z.R.; Lin, H.H.; Han, L.Y.; Jiang, L.; Chen, X.; Chen, Y.Z. PROFEAT: A web server for computing structural and physicochemical features of proteins and peptides from amino acid sequence. *Nucleic Acids Res.* **2006**, *34*, W32–W37. [CrossRef] [PubMed]
43. Shen, H.-B.; Chou, K.-C. PseAAC: A flexible web server for generating various kinds of protein pseudo amino acid composition. *Anal. Biochem.* **2008**, *373*, 386–388. [CrossRef] [PubMed]
44. Cao, D.-S.; Xu, Q.-S.; Liang, Y.-Z. Propy: A tool to generate various modes of Chou's PseAAC. *Bioinformatics* **2013**, *29*, 960–962. [CrossRef] [PubMed]
45. Du, P.; Wang, X.; Xu, C.; Gao, Y. PseAAC-Builder: A cross-platform stand-alone program for generating various special Chou's pseudo-amino acid compositions. *Anal. Biochem.* **2012**, *425*, 117–119. [CrossRef] [PubMed]
46. Du, P.; Gu, S.; Jiao, Y. PseAAC-General: Fast building various modes of general form of Chou's pseudo-amino acid composition for large-scale protein datasets. *Int. J. Mol. Sci.* **2014**, *15*, 3495–3506. [CrossRef] [PubMed]
47. Xiao, N.; Cao, D.-S.; Zhu, M.-F.; Xu, Q.-S. protr/ProtrWeb: R package and web server for generating various numerical representation schemes of protein sequences. *Bioinformatics* **2015**, *31*, 1857–1859. [CrossRef] [PubMed]
48. Chen, W.; Lei, T.-Y.; Jin, D.-C.; Lin, H.; Chou, K.-C. PseKNC: A flexible web server for generating pseudo K-tuple nucleotide composition. *Anal. Biochem.* **2014**, *456*, 53–60. [CrossRef] [PubMed]
49. Chen, W.; Zhang, X.; Brooker, J.; Lin, H.; Zhang, L.; Chou, K.-C. PseKNC-General: A cross-platform package for generating various modes of pseudo nucleotide compositions. *Bioinformatics* **2015**, *31*, 119–120. [CrossRef] [PubMed]

50. Liu, B.; Liu, F.; Fang, L.; Wang, X.; Chou, K.-C. repDNA: A Python package to generate various modes of feature vectors for DNA sequences by incorporating user-defined physicochemical properties and sequence-order effects. *Bioinformatics* **2015**, *31*, 1307–1309. [CrossRef] [PubMed]
51. Liu, B.; Liu, F.; Fang, L.; Wang, X.; Chou, K.-C. repRNA: A web server for generating various feature vectors of RNA sequences. *Mol. Genet. Genom.* **2016**, *291*, 473–481. [CrossRef] [PubMed]
52. Liu, B.; Liu, F.; Wang, X.; Chen, J.; Fang, L.; Chou, K.-C. Pse-in-One: A web server for generating various modes of pseudo components of DNA, RNA, and protein sequences. *Nucleic Acids Res.* **2015**, *43*, W65–W71. [CrossRef] [PubMed]
53. Li, T.; Du, P.; Xu, N. Identifying human kinase-specific protein phosphorylation sites by integrating heterogeneous information from various sources. *PLoS ONE* **2010**, *5*, e15411. [CrossRef] [PubMed]
54. Chen, Q.-Y.; Tang, J.; Du, P.-F. Predicting protein lysine phosphoglycerylation sites by hybridizing many sequence based features. *Mol. Biosyst.* **2017**, *13*, 874–882. [CrossRef] [PubMed]
55. Lei, G.-C.; Tang, J.; Du, P.-F. Predicting *S*-sulfenylation Sites Using Physicochemical Properties Differences. *Lett. Org. Chem.* **2017**, *14*, 665–672. [CrossRef]
56. Steinbiss, S.; Gremme, G.; Schärfer, C.; Mader, M.; Kurtz, S. AnnotationSketch: A genome annotation drawing library. *Bioinformatics* **2009**, *25*, 533–534. [CrossRef] [PubMed]
57. Jia, J.; Liu, Z.; Xiao, X.; Liu, B.; Chou, K.-C. iCar-PseCp: Identify carbonylation sites in proteins by Monte Carlo sampling and incorporating sequence coupled effects into general PseAAC. *Oncotarget* **2016**, *7*, 34558–34570. [CrossRef] [PubMed]
58. Qiu, W.-R.; Xiao, X.; Lin, W.-Z.; Chou, K.-C. iMethyl-PseAAC: Identification of protein methylation sites via a pseudo amino acid composition approach. *Biomed. Res. Int.* **2014**, *2014*, 947416. [CrossRef] [PubMed]
59. Liu, B.; Xu, J.; Lan, X.; Xu, R.; Zhou, J.; Wang, X.; Chou, K.-C. iDNA-Prot|dis: Identifying DNA-binding proteins by incorporating amino acid distance-pairs and reduced alphabet profile into the general pseudo amino acid composition. *PLoS ONE* **2014**, *9*, e106691. [CrossRef] [PubMed]
60. Xu, Y.; Wen, X.; Wen, L.-S.; Wu, L.-Y.; Deng, N.-Y.; Chou, K.-C. iNitro-Tyr: Prediction of nitrotyrosine sites in proteins with general pseudo amino acid composition. *PLoS ONE* **2014**, *9*, e105018. [CrossRef] [PubMed]
61. Chang, C.-C.; Lin, C.-J. LIBSVM: A Library for Support Vector Machines. *ACM Trans. Intell. Syst. Technol.* **2011**, *2*, 27. [CrossRef]