An open-source framework for evaluating MS/MS score functions: *"pluggable scoring"* in X! TANDEM

Brendan MacLean[2], Jimmy Eng[1], Ronald C. Beavis[3], Martin McIntosh[1]

[1]*Computational Proteomics Laboratory, Fred Hutchinson Cancer Research Center, Seattle, WA USA*

[2]*LabKey Software, LLC, Seattle, WA USA*

[3]*Beavis Informatics LTD, Winnipeg, MB Canada*

FRED HUTCHINSON CANCER RESEARCH CENTER
A LIFE OF SCIENCE

TP 655, ASMS 2006

Overview

The X! TANDEM software tool has been adapted to allow novel score functions to be "plugged in". The goal is to facilitate and foster the development, implementation, and evaluation of better ways of identifying peptides by tandem mass spectrometry database searching. This is achieved by modifying TANDEM such that core infrastructure is separated from the actual score function module, enabling the scoring class to be overridden. We present a general introduction to the software modifications that enable pluggable scoring, supply example score function modules, and discuss plans for future work.

Introduction

Peptide and protein identification via tandem mass spectrometry and sequence database searching has become ubiquitous in the realm of proteomics. As the field of proteomics grows and attracts a larger computational audience, there is an ever increasing emphasis on developing more accurate and sensitive computational methods to identify the constituent proteins in a sample. Most of this research has focused on MS/MS database searching and more specifically on how to better score a match between a query spectrum and candidate peptides from a sequence database.

The most prevalent MS/MS database search tools in use today are commercial tools that are typically proprietary and not amenable to user modification. The recent availability of open source MS/MS search tools such as TANDEM[1], OMSSA[2], and ProbID[3] allows researchers the opportunity to understand the mechanisms behind how they work and make modifications to them as needed. For this project, we focused on adding pluggable scoring to the TANDEM software as we found that the software is well-architected, contains many rich features, has rapidly gained traction within the proteomics community, and allows us to be compatible with and take advantage of the rich and growing set of tools of the GPM[4] project. By extending TANDEM to make the score function pluggable, we hope to lower the barrier to development and evaluation of new scoring algorithms with the goal of fostering research in the realm of mass spectrometry-based identifications.

Results

"Native": The native TANDEM score algorithm is detailed in the "`mscore_tandem.cpp`" and "`mscore_tandem.h`" files of the TANDEM project. Our plug-in work separated the actual scoring mechanisms from the supporting framework and placed the native score function in a separate class file.

"S-score": This score plug-in module calculates a score using the sum of the logs of the intensities of matched peaks divided by the square root of the sequence length. This score algorithm is implemented in the two files "`mscore_s.cpp`" and "`mscore_s.h`" and is meant to be an easy to understand, simple implementation of a scoring algorithm that can be used as a template for developing new score plug-ins.

"K-score": The second score plug-in that we have implemented is referred to as the "K-score" algorithm. The associated score plug-in files are named "`mscore_k.cpp`" and "`mscore_k.h`". This plug-in implements the spectral processing and scoring described in [5], which effectively is just a dot product on a processed spectrum. Notably this score plug-in has PeptideProphet[6] support although "Native" will also have this support soon.

The specification of which scoring algorithm to apply to a search is accomplished using the following TANDEM API parameter:

```
<note type="input" label="scoring, algorithm">VALUE</note>
```

"`s-score`" and "`k-score`" are the appropriate values to specify here in order to apply the "S-score" or "K-score" algorithms, respectively. TANDEM's native algorithm is the default scoring algorithm applied; for this case either leave out this API entry or explicitly use the value "`default`".

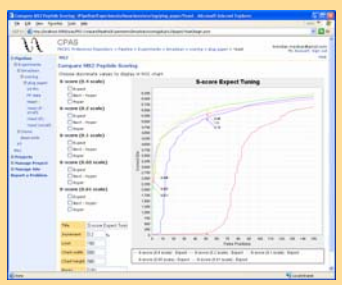
It should be noted that these two additional score plug-in modules are just our initial attempts at putting together example modules. With this framework in place to implement and evaluate scoring algorithms, both in the modifications to TANDEM to more easily support pluggable scoring and in the functionality built into CPAS (described to the right), we plan to now investigate developing novel algorithms to more sensitively identify peptides and proteins interrogated with tandem mass spectrometry.

CPAS Analysis Framework

To help evaluate current and future score functions, a "Compare MS2 Peptide Scoring" module was built into CPAS[7] to facilitate the comparison of search performance.

The comparison module chooses a number (*n*) of top scoring peptides for increasing *n*, computes the number of correct and incorrect peptides among them, denoted Y(*n*) and X(*n*), respectively, and characterizes the overall accuracy of each scoring algorithm by plotting Y(*n*) versus X (*n*) for all *n*. Correctness/incorrectness is simply assumed based on a forward + large reverse or randomized database search.

This "Compare MS2 Peptide Scoring" module is implemented in CPAS and part of the available source code, but this functionality is not currently compiled into the Windows graphical installer distribution of CPAS 1.4.



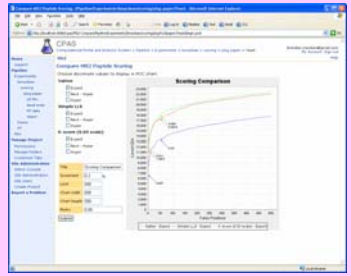
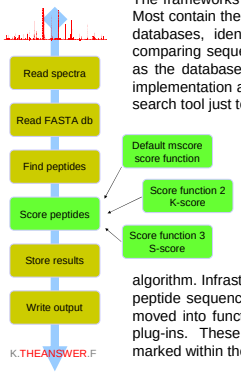


Figure 1.
This plot displays the performance of the S-score and K-score plug-in modules against the native TANDEM score. The score parameter used for all three analyses was the expectation value, which is a statistical measure that the particular score is expected to be achieved merely by chance. Although other comparison parameters are available, using the expectation value allows the score comparisons to be consistently applied across the different score functions.

Figure 2.
This plot displays the performance of the K-score plug-in modules as a scaling parameter (`m_dScale`) is varied. This parameter affects the granularity of the score histogram that TANDEM uses to calculate the expectation values of the search results. This figure shows that there are ancillary parameters that could affect optimal scoring and that these parameters can be elucidated using the "Compare MS2 Peptide Scoring" module of CPAS.

Methods



The frameworks for various MS/MS database search tools are conceptually very similar. Most contain the same core component steps: reading in input spectra, parsing sequence databases, identifying candidate peptides of the same mass as the input spectra, comparing sequence against spectrum to generate a score, storing relevant peptide hits as the database is being searched, and, lastly, writing out the search results. So the implementation and testing of a new score function typically requires developing an entire search tool just to investigate one (albeit very important) subcomponent.

To facilitate the development and evaluation of novel score functions, we modified the TANDEM program to allow scoring modules to be easily pluggable. The modifications enable adding new (and modifying existing) score functions without requiring any modifications to the rest of the source code.

The modifications to the TANDEM source involve first separating out general scoring infrastructure from the actual native TANDEM score algorithm. Infrastructure for scoring functions, such as a state machine for choosing which peptide sequences to score, was left in the base class. Specific scoring decisions were moved into functions that were then made virtual so they could be overridden by other plug-ins. These virtual functions, which represent the true "Pluggable Scoring API," are marked within the mscore.h file in the TANDEM project (see insert to the right).

The source code modifications also include the creation of a generalized plug-in manager. The plug-in manager maps a plug-in type to a set of named plug-ins; see mplugin.h and mplugin.cpp in the source distribution for details. The creation of a new score plug-in involves implementation of a single C++ source file and single corresponding header file. Scoring modules can be added or removed simply by adding or removing the pair of .cpp/.h files from the project and recompiling. No other changes are required, either to the Makefile or the other source files, to add/remove these scoring plug-ins.

The modifications made to TANDEM, as well as TANDEM's native hyperscore score function (referred to as the mscore function) are available as part of the default TANDEM distribution. The 'S-score' module, which is a score function implementing simple spectral processing + scoring and intended to be used as a template for developing new plug-ins, along with other current and future scoring modules, will be available from the CPL website at *proteomics.fhcrc.org* pending publication.

mscore.h

```
/*
  The following section represents the pluggable scoring API devised and implemented by Brendan MacLean. It
  allows developers to simply add new scoring systems to X! Tandem for any purpose.  Please do not modify
  without considering carefully how changes might impact external scoring plug-ins.
*/
  virtual bool load_param(XmlParameter &_x); // allows score object to issue warnings/set vars based on xml
  virtual bool precondition(mspectrum &_s); // called before spectrum conditioning
  virtual bool add_details(mspectrum &_s);
  virtual bool add_mi(mspectrum &_s);
  virtual void prescore(const size_t _i); // called before scoring
  virtual float score(const size_t _i);
/*
  If you have no need for scoring plug-ins other than mscore_tandem, making this function non-virtual can
  yield a 10% perf improvement.
*/
#ifdef PLUGGABLE_SCORING
  virtual unsigned long mconvert(double _m, const long _c); // convert mass to integer ion m/z for mi vector
#else
  unsigned long mconvert(double _m, const long _c); // convert mass to integer ion m/z for mi vector
#endif
  virtual double hfactor(long _l); // hyper scoring factor from number of ions matched
  virtual double sfactor(); // factor applied to final convolution score
  virtual float mconvert(float _h); // convert hyper score to histogram score
  virtual void report_score(char* _buff, float _h); // format hyper score for output
  virtual bool clear();
protected:
  virtual double dot(unsigned long *_v) = 0; // this is where the real scoring happens
/*   End pluggable scoring API. */
```

Conclusions

- We implemented pluggable scoring by modifying TANDEM's score search program to facilitate the addition of new scoring algorithms without requiring modifications to the rest of the TANDEM source code
- We present example score function plug-in modules that can be used as templates to develop new scoring algorithms
- We demonstrate a framework for evaluating/comparing the performance of the various plug-ins within the CPAS analysis system
- Future work includes plans to investigate the development of more accurate and sensitive scoring algorithms; hopefully this work encourages other computational researchers to do the same
- This work builds on the TANDEM program and allows us to take advantage of the rich and growing open source development community fostered by the GPM project

References

1. Craig R and Beavis RC. *Bioinformatics*. 2004 Jun 12; 20(9):1466-7.
2. Geer LY *et al*. *Journal of Proteome Research*. 2004 Sep-Oct;3(5):958-64.
3. Zhang N *et al*. *Proteomics*. 2002 Oct; 2(10):1406-12.
4. Craig R *et al*. *Journal of Proteome Research*. 2004 Nov-Dec;3(6):1234-42
5. Keller A *et al*. *Molecular Systems Biology*. 2005 Aug 2; Msb4100024.
6. Nesvizhskii *et al*. *Analytical Chemistry*. 2003 Sep 1;75(17):4646-58.
7. Rauch A *et al*. *Journal of Proteome Research*. 2006; 5(1), 112-121.

This work was funded by NCI contract 23XS144A