Using MSFragger for ultrafast database searching

Alexey I. Nesvizhskii (nesvi@med.umich.edu)
Nesvizhskii Lab / University of Michigan
Andy T Kong
Nesvizhskii Lab / University of Michigan
Felipe V. Leprevost
Nesvizhskii Lab / University of Michigan
Dmitry M. Avtonomov
Nesvizhskii Lab / University of Michigan
Dattatreya Mellacheruvu
Nesvizhskii Lab / University of Michigan

Method Article

Keywords: Mass spectrometry, Proteome informatics, Proteomics, Software

Posted Date: April 10th, 2017

DOI: https://doi.org/10.1038/protex.2017.032

Abstract

Database search has long been the preferred method for peptide identification in shotgun proteomics. While a wide range of database search tools exist and excel for conventional workflows, they are too slow for open database searching. To make open searching practical for routine analysis, we developed a novel database search tool called MSFragger that is over 100 times faster than most existing tools. This protocol demonstrates how to perform database searching using MSFragger. Up-to-date downloads of MSFragger and associated documentation can be found at "nesvilab.org/software.html":http://nesvilab.org/software.html.

Introduction

MSFragger is an ultrafast database search tool for peptide identifications in mass spectrometry-based proteomics. It differs from conventional search engines by computing similarity scores in a fragment-centric fashion using a theoretical fragment index of candidate peptides. The speed of MSFragger makes it particularly suitable for ‘open’ database searches, where the precursor mass tolerance is set to hundreds of Daltons, for the identification of modified peptides. MSFragger is implemented in the cross-platform Java programming language and is compatible with standard proteomics file formats such as MGF/mzXML/mzML/pepXML. This protocol contains the necessary procedures for running the initial release \(20170103.0) of MSFragger. Up-to-date downloads of the software and manuals can be found at "nesvilab.org/software.html":http://nesvilab.org/software.html.

Equipment

Computer Hardware requirements The processor requirements of MSFragger depends on the complexity of your search \(and your patience to wait for search results). For an open search \(500Da precursor mass window) using a tryptic digest of the human proteome, a single processor core can search roughly 40,000 MS/MS spectra in under an hour. MSFragger scales well with the number of processor cores and runtimes of under 2 minutes per file have been achieved using a 28-core workstation. A desktop workstation with a quad core processor is sufficient for most simple workflows. MSFragger requires substantial amounts of memory due to its in-memory fragment index. While MSFragger can operate with less memory than needed to store the fragment index, it will cause index fragmentation where it breaks the search into multiple passes, searching each input file against a small segment of the index at a time \(which greatly increases the runtime). For the human Uniprot protein database with reversed decoys, approximately 3700 MB of memory is needed to prevent index fragmentation. The actual size of the fragment index is substantially lower \(MSFragger uses a very conservative estimate of the available free memory to avoid out of memory situations). Specifying common modifications may boost memory requirements to 6 GB. Semi-tryptic, non-enzymatic, and phospho searches may take tens of gigabytes to avoid fragmented searches. Limiting the range of peptide lengths can reduce the search space and reduce memory consumption in such cases. While fragment index fragmentation is undesirable, it may be unavoidable in certain instances. We recommend at least 8GB of memory for workflows involving standard tryptic digestions. **Operating System requirements** MSFragger has been tested on Mac OS X, Windows 7, and a number of Linux distributions. Note that a 64-bit

operating system is required to access more than 4GB of memory. **Java requirements** MSFragger is written using Java 1.8 and requires the Java 8 Runtime Environment. We recommend the Oracle Java 8 Runtime \(download and installation instructions are available at "www.java.com":http://www.java.com).

Procedure

Preparing Input Files Mass spectrometry data must first be converted to one of the supported MS/MS input formats of MGF, mzXML, or mzML. A popular option for converting from vendor file inputs and between various input formats is Proteowizard \("proteowizard.sourceforge.net":http://proteowizard.sourceforge.net). MSFragger determines the appropriate data parser to use based on the file extension \(.mgf for MGF, .mzXML for mzXML, and .mzML for mzML) and does not make inferences from file contents \(i.e. naming a mzML file with the .mzXML extension will lead to unpredictable results or crashes). The protein database must be supplied in FASTA format. MSFragger does not have the capability to generate decoys internally so they must be generated externally and appended to the protein database before running MSFragger. **Configuring MSFragger** Extract the MSFragger.jar into your working directory along with the sample configuration file called fragger.params. MSFragger is configured using a text parameters file. The parameters file is passed as the first argument to MSFragger and has no restrictions on names or file extensions \(so one might want to name their configuration files to be more descriptive such as Uniprot_open_withmods.txt) after editing the parameters file for a particular analysis. Parameter names are given left of the equal sign and parameter values are given to the right \(e.g. num_threads = 4). White spaces are trimmed from the ends of each value by MSFragger. All text to the right of \(and including) the # sign of each line is discarded so # can be used for comments in the parameters file. A detailed explanation of each parameter can be found in "this PDF":http://www.nature.com/protocolexchange/system/uploads/5227/original/MSFraggerParamGuide.pdf?1488158913. **Performance Considerations for Batch Processing** MSFragger allows multiple MS/MS input files to be processed in a batch. Passing multiple files to MSFragger at once allows MSFragger to reuse the fragment index for subsequent MS/MS run. This is particularly important for narrow window searches which may only take fractions of a second. On computers or compute clusters with many processor cores, we highly recommended that MSFragger is set to process files sequentially with all available processor cores rather than running multiple instances of MSFragger in parallel \(assigning a smaller number of cores to each). This reduces initialization times and allows the fragment index to be re-used, at the same time reducing overall memory requirements. **Launching MSFragger** Ensure that you have placed MSFragger.jar in your working directory and have modified the parameters file to reference your protein database. MSFragger generates auxiliary files during database search so it is critical that **MSFragger must have write access to the directories containing the protein database AND the MS/MS data files**. Determine the amount of system memory available that you would like to make available to MSFragger. This will be specified by the Java maximum heap size parameter -Xmx \(e.g. -Xmx3700M for 3700 MB or -Xmx8G for 8GB). MSFragger takes the first argument as the input parameters file, followed by a list of one or more MS/MS data files. Examples: java -Xmx8G -jar MSFragger.jar fragger.params HeLa_run1.mzML HeLa_run2.mzML java -Xmx8G -jar MSFragger.jar fragger.params *.mzML The **-Xmx flag is very important** to ensure that MSFragger has access to sufficient memory to efficiently perform the search as the default max heap setting in Java is 1/4 of total system memory \(which is insufficient for optimal performance). We recommend that you can allocate a minimum of 4G or 6G for standard tryptic digestions.

Anticipated Results

Expected Behavior The first time running MSFragger on a new protein database or set of search parameters with a given database, it will first perform an in-silico digestion, create, and cache the peptide index \(in .pepindex files adjacent to where the FASTA database is stored). These pepindex files can be safely removed at any time and should be removed to free up disk space when a set of search parameters is no longer used \(MSFragger will automatically re-generate the index as needed). The process begins with filtering and in-silico digestion subject to the digestion parameters. See figure in Figures section. Followed by peptide sorting and de-duplication. The non-redundant set of peptides are then evaluated to generate the set of variably modified peptides \(based on the specified variable modifications) which are then sorted by mass and stored. See figure in Figures section. After peptide index generation is complete \(or is read from disk in the below screenshot). MSFragger selects the fragment index bin width to use and estimates the memory available for fragment index storage based on the available memory \(in this case, 8GB of memory was made available to the Java Virtual Machine, of which MSFragger estimates that 4976.67MB can be safely reserved for fragment index operations). It then computes the number of theoretical fragments to be generated for the entire index, the number of slices or iterations \(in multi-pass searches when there is insufficient memory), and the total amount of memory represented by the entire fragment index. The fragment index is then generated, and a time is reported for the index generation time \(at the end of each Operating on slice 1 of X: line, 4770 ms below). If the maximum fragment slice size is very small compared to your desired amount of system memory or the number of slices is unexpectedly high, double check that the -Xmx flag is correctly set. See figure in Figures section. Search begins and the current file is reported, along with the time needed to read and pre-process the MS/MS data, along with current search progress. See figure in Figures section. At the completion of the search, a completed time is reported, and the results are written to disk in the same folder as the MS/MS data \(if they are not in the same folder as your working directory). Note that there is a current bug that causes MSFragger to incorrectly display the average rate of matching at the conclusion of the run \(although the total time can be divided by the total number of spectra to calculate this value). **Output Files** .fragtmp files In cases of fragment index fragmentation \(in limited memory scenarios), MSFragger will iteratively load each MS/MS run and search loaded spectra against the current index slice before working on the next index slice. The partial search results are then stored in these .fragtmp files. In the event that MSFragger is terminated in the middle of a search, it will recover its partial results using these files. At the end of the last index slice, MSFragger will read all such .fragtmp files and generate an aggregated results file \(identical to one that would be generated if it had the memory to search against all peptides in a single pass). These .fragtmp files are then automatically deleted. These can be safely removed if you no longer wish to continue an aborted search or if MSFragger somehow fails to remove them at the conclusion of a successful search. Location: Same directory as MS/MS files .pepindex files MSFragger stores the computed peptide index in .pepindex files adjacent to the protein database files to remove the need to re-compute the index if search parameters are unchanged in subsequent runs. These .pepindex indices can be safely removed and MSFragger will re-compute the index again at runtime if needed. Location: Same directory as protein database Results Files \(eg. .pep.xml, .tsv) These are the pepXML or TSV output files containing the peptide identifications. The file extension is specified in the search parameters so specifying a .pep.xml extension with output_format = tsv will output .pep.xml files with TSV content. Location: Same directory as MS/MS files **Interpretation of Output** For pepXML outputs, these can be used for downstream

processing using PeptideProphet in TPP directly. For viewing of results or conversion to other peptide identification result formats for use in other pipelines or tools that do not support pepXML, we recommend first converting to the mzIdentML format using the tool idconvert as part of the ProteoWizard package. The pepXML generated by MSFragger validates against v 1.18 of the pepXML schema and should be compatible with any downstream tools supporting the pepXML format. The order of the output fields in the TSV file produced by MSFragger are: ScanID, Precursor neutral mass \(Da), Retention time \(minutes), Precursor charge, Hit rank, Peptide Sequence, Upstream Amino Acid, Downstream Amino Acid, Protein, Matched fragment ions, Total possible number of matched theoretical fragment ions, Neutral mass of peptide \(including any variable modifications) \(Da), Mass difference, Number of tryptic termini, Number of missed cleavages, Variable modifications detected \(starts with M, separated by |, formated as position,mass), Hyperscore, Next score, Intercept of expectation model \(expectation in log space), Slope of expectation model \(expectation in log space)

Figures

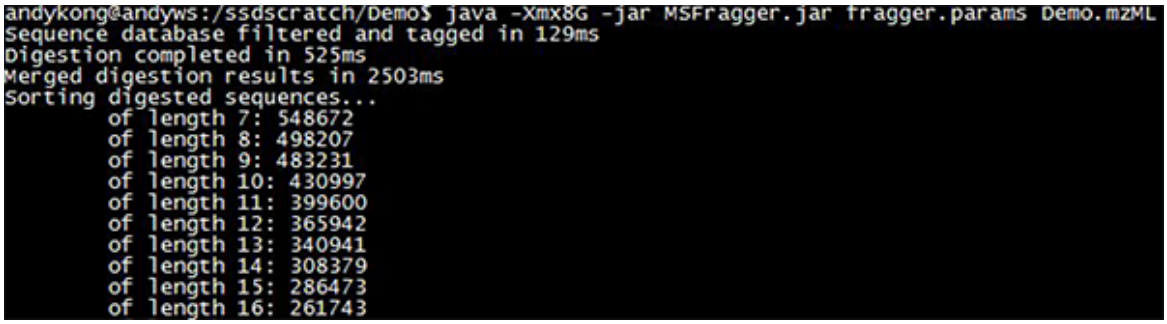

Figure 1

In-silico digestion phase

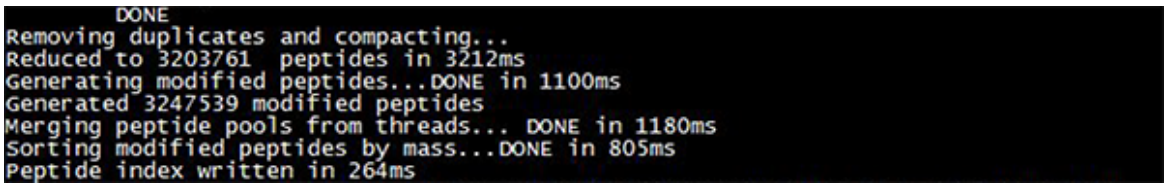

Figure 2

Peptide index generation

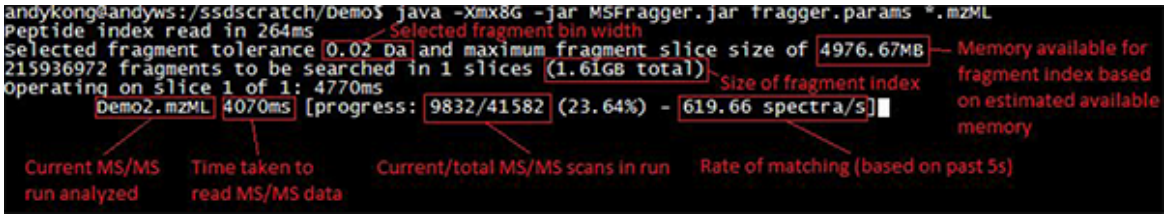

Figure 3

Main search screen

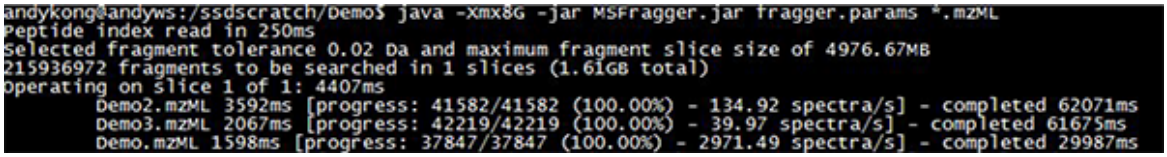

Figure 4

MSFragger in batch mode

Supplementary Files

This is a list of supplementary files associated with this preprint. Click to download.

- supplement0.pdf