MMseqs2 User Guide

Martin Steinegger Milot Mirdita Lars von den Driesch
Clovis Galiez Johannes Söding

Contents

Summary

MMseqs2 (Many-against-Many searching) is a software suite to search and cluster huge sequence sets. MMseqs2 is open source GPL-licensed software implemented in C++ for Linux, Mac OS and Windows. The software is designed to run on multiple cores and servers and exhibits very good scalability. MMseqs2 reaches the same sensitivity as BLAST magnitude faster and which can also perform profile searches like PSI-BLAST but also 400 times faster.

At the core of MMseqs2 are two modules for the comparison of two sequence sets with each other - the prefiltering and the alignment modules. The first, prefiltering module computes the similarities between all sequences in one query database with all sequences a target database based on a very fast and sensitive k-mer matching stage followed by an ungapped alignment. The alignment module implements an vectorized Smith-Waterman alignment of all sequences that pass a cut-off for the ungapped alignment score in the first module. Both modules are parallelized to use all cores of a computer to full capacity. Due to its unparalleled combination of speed and sensitivity, searches of all predicted ORFs in large metagenomics data sets through the entire UniProtKB or NCBI-NR databases are feasible. This allows for assigning to functional clusters and taxonomic clades many reads that are too diverged to be mappable by current software.

MMseqs2 clustering module can cluster sequence sets efficiently into groups of similar sequences. It takes as input the similarity graph obtained from the comparison of the sequence set with itself in the prefiltering and alignment modules. MMseqs2 further supports an updating mode in which sequences can be added to an existing clustering with stable cluster identifiers and without the need to recluster the entire sequence set. We are using MMseqs2 to regularly update versions of the UniProtKB database clustered down to 30% sequence similarity threshold. This database is available at uniclust.mmseqs.com.

System Requirements

MMseqs2 runs on modern UNIX operating systems and is tested on Linux and OSX. Additionally, we are providing a preview version for Windows.

The alignment and prefiltering modules are using with SSE4.1 (or optionally AVX2) and OpenMP, i.e. MMseqs2 can take advantage of multicore computers.

When searching large databases, MMseqs2 may need a lot main memory (see section memory requirements). We offer an option for limiting the memory usage at the cost of longer runtimes. The database is split into chunks and the program only holds one chunk in memory at any time. For clustering large databases containing tens of millions of sequences, you should provide enough free disc space (~500 GB). In section Optimizing Sensitivity and Consumption of Resources, we will discuss the runtime, memory and disc space consumption of MMseqs2 and how to reduce resource requirements for large databases.

Check system requirements

To check if MMseqs2 supports your system execute the following commands, depending on your operating system: #### Linux

```
[[ $(uname -m) == "x86_64" ]] && echo "64bit Supported" || echo "64bit Unsupported"
cat /proc/cpuinfo | grep -c sse4_1 > /dev/null && echo "SSE4.1 Supported" || echo "SSE4.1 Un
cat /proc/cpuinfo | grep -c avx2 > /dev/null && echo "AVX2 Supported" || echo "AVX2 Unsuppor
```

MacOS

```
[[ $(uname -m) == "x86_64" ]] && echo "64bit Supported" || echo "64bit Unsupported"
sysctl -a | grep machdep.cpu.features | grep -c SSE4.1 > /dev/null && echo "SSE4.1 Supporte
sysctl -a | grep machdep.cpu.leaf7_features | grep -c AVX2 > /dev/null && echo "AVX2 Support
```

Windows The `mmseqs.bat` script will print a message if its run on an unsupported system. On a supported system, it will execute the correct MMseqs2 version and forward all parameters.

Installation

MMseqs2 can be installed by downloading a statically compiled version, compiling the from source, using Homebrew or Docker.

Install static Linux version

The following command will download the lastest MMseqs2 version, extract it and set the PATH variable.

Linux If your computer supports AVX2 use:

```
wget https://mmseqs.com/latest/mmseqs-static_avx2.tar.gz
tar xvzf mmseqs-static_avx2.tar.gz
export PATH=$(pwd)/mmseqs/bin/:$PATH
```

If your computer supports SSE4.1 use:

```
wget https://mmseqs.com/latest/mmseqs-static_sse41.tar.gz
tar xvzf mmseqs-static_sse41.tar.gz
export PATH=$(pwd)/mmseqs/bin/:$PATH
```

Mac If your computer supports AVX2 use:

```
wget https://mmseqs.com/latest/mmseqs-osx-static_avx2.tar.gz
tar xvzf mmseqs-osx-static_avx2.tar.gz
export PATH=$(pwd)/mmseqs/bin/:$PATH
```

If your computer supports SSE4.1 use:

```
wget https://mmseqs.com/latest/mmseqs-osx-static_sse41.tar.gz
tar xvzf mmseqs-osx-static_sse41.tar.gz
export PATH=$(pwd)/mmseqs/bin/:$PATH
```

Windows (preview) The latest version is always available on:

```
https://mmseqs.com/latest/mmseqs-win64.zip
```

Download and unzip it at a convenient location. Inside you will find the `mmseqs.bat` wrapper script, which should be used to substitute all calls to `mmseqs` in the remainder of this document, and a `bin` folder with all dependencies of the MMseqs2 Windows version. Please always keep the `mmseqs.bat` script one folder above the `bin` folder, or it will not be able to correctly identify its dependencies anymore.

The windows build also contains both the SSE4.1 and the AVX2 version. The `mmseqs.bat` script will automatically choose the correct one.

Compile from source

Compiling MMseqs2 from source has the advantage that it will be optimized to the specific system, which should improve its performance. To compile MMseqs2 `git`, `g++` (4.6 or higher) and `cmake` (3.0 or higher) are needed. Afterwards, the MMseqs2 binary will be located in `build/bin/`.

```
git clone https://github.com/soedinglab/MMseqs2.git
cd MMseqs2
mkdir build
cd build
cmake -DCMAKE_BUILD_TYPE=RELEASE -DCMAKE_INSTALL_PREFIX=. ..
make
make install
export PATH=$(pwd)/bin/:$PATH
```

:exclamation: On MacOS, please install the `gcc@7 zlib bzip2 vim cmake` packages from Homebrew, if you want to compile MMseqs2. The default MacOS `clang` compiler does not support OpenMP and MMseqs2 will not be able to run multithreaded. Use the following cmake call:

```
CXX="$(brew --prefix)/bin/g++-7" cmake -DCMAKE_BUILD_TYPE=RELEASE -DCMAKE_INSTALL_PREFIX=. .
```

Windows The windows build process is more involved due to MMseqs2's dependency on an installed shell. We use the Cygwin environment and Busybox to provide all necessary dependencies and bundle them all together. If you want to compile MMseqs2 on your own, install the following packages from Cygwin:

```
bash xxd cmake make gcc-g++ zlib-devel libbz2-devel busybox-standalone binutils
```

Afterwards, use a workflow similar to the `util/build_windows.sh` script to build MMseqs2 on Windows.

Install with Homebrew

You can install the latest stable version of MMseqs2 for Mac OS through Homebrew by executing the following:

```
brew install mmseqs2
```

MMseqs2 built from the latest git commit, can be installed with the following command:

```
brew install https://raw.githubusercontent.com/soedinglab/mmseqs2/master/Formula/mmseqs2.rb
```

This will also automatically install the bash completion (you might have to execute `brew install bash-completion` first). This will also work for Linuxbrew.

Use the Docker image

You can pull the official docker image by running:

```
docker pull soedinglab/mmseqs2
```

If you want to build the docker image from the git repository, execute:

```
git clone https://github.com/soedinglab/MMseqs2.git
cd MMseqs2
docker build -t mmseqs2 .
```

Use the BASH command completion

MMseqs comes with a bash command and parameter auto completion by pressing tab. The bash completion for subcommands and parameters can be installed by adding the following lines to your $HOME/.bash_profile:

```
if [ -f /Path to MMseqs2/util/bash-completion.sh ]; then
    source /Path to MMseqs2/util/bash-completion.sh
fi
```

Getting Started

Here we explain how to run a search for sequences matches in the query database against a target database and how to cluster a sequence database. Test data (a query and a target database for the sequence search and a database for the clustering) are stored in the `examples` folder.

Search

Before searching, you need to convert your FASTA file containing query sequences and target sequences into a sequence DB. You can use the query database `examples/QUERY.fasta` and target database `examples/DB.fasta` to test the search workflow:

```
$ mmseqs createdb examples/QUERY.fasta queryDB
$ mmseqs createdb examples/DB.fasta targetDB
```

These calls should generates five files each, e.g. queryDB, queryDB_h and its corresponding index file queryDB.index, queryDB_h.index and queryDB.lookup from the FASTA QUERY.fasta input sequences.

The queryDB and queryDB.index files contain the amino acid sequences, while the queryDB_h and queryDB_h.index file contain the FASTA headers. The queryDB.lookup file contains a list of tab separated fields that map from the internal identifier to the FASTA identifiers.

Important: createdb splits long sequences into multiple separate entries automatically. This avoids excessive resource requirements for later steps. The default value is to split sequences after 32000 residues. The identifiers of the new entries are suffixed with _0 to _(n-1) for N splits.

For the next step, an index file of the targetDB is computed for a fast read in. It is recommend to compute the index if the targetDB is reused for several searches.

```
$ mmseqs createindex targetDB tmp
```

This call will create a targetDB.sk6 file. In this file extension the letter s indicates the use of spaced k-mers and the k6 shows the k-mer size of 6.

Then generate a directory for temporary files. MMseqs2 can produce a high IO on the file system. It is recommended to create this temporary folder on a local drive.

```
$ mkdir tmp
```

Please ensure that in case of large input databases tmp provides enough free space. For the disc space requirements, see the section Disc Space.

The alignment consists of two steps the prefilter and alignment. To run the search, type:

```
$ mmseqs search queryDB targetDB resultDB tmp
```

Search as standard does compute the score only. If you need the alignment information add the option "-a".

Then, convert the result database into a BLAST tab formatted file (option -m 8 in legacy blast, -outfmt 6 in blast+):

```
$ mmseqs convertalis queryDB targetDB resultDB resultDB.m8
```

The file is formatted as a tab-separated list with 12 columns: (1,2) identifiers for query and target sequences/profiles, (3) sequence identity, (4) alignment length, (5) number of mismatches, (6) number of gap openings, (7-8, 9-10) domain start and end-position in query and in target, (11) E-value, and (12) bit score.

Read more about searching here.

Clustering

Before clustering, convert your FASTA database into the MMseqs database (DB) format:

```
$ mmseqs createdb examples/DB.fasta DB
```

Then, generate a directory for tmp files:

```
$ mkdir tmp
```

Please ensure that in case of large input databases `tmp` provides enough free space. For the disc space requirements, see the section Disk space.

Run the clustering of your database `DB` by executing the following command. MMseqs2 will return the result database files `DB_clu`, `DB_clu.index`:

```
$ mmseqs cluster DB DB_clu tmp
```

To generate a TSV formatted output file from the output file, type:

```
$ mmseqs createtsv DB DB DB_clu DB_clu.tsv
```

You can adjust the sequence identity threshold with `--min-seq-id` and the alignment coverage with `-c` and `--cov-mode`. MMseqs2 will set the sensitivity parameters automatic based on target sequence identity (`--min-seq-id`), if it is not already specified through the `-s` or `--k-score` parameters.

Sequence information can be added by using createseqfiledb and result2flat can produce a result.

```
$ mmseqs createseqfiledb DB DB_clu DB_clu_seq
$ mmseqs result2flat DB DB DB_clu_seq DB_clu_seq.fasta
```

Read more about clustering here.

Linclust

Linclust is a clustering in linear time. It is magnitudes faster but a bit less sensitive than clustering.

Before clustering, convert your FASTA database into the MMseqs database (DB) format:

```
$ mmseqs createdb examples/DB.fasta DB
```

Then, generate a directory for tmp files:

```
$ mkdir tmp
```

To run linclust the clustering of your database `DB` by executing the following command. The result database follows the same format as the clustering format:

```
$ mmseqs linclust DB DB_clu tmp
```

To extract the representative sequences from the clustering result call:

```
mmseqs result2repseq DB DB_clu DB_clu_rep
mmseqs result2flat DB DB DB_clu_rep DB_clu_rep.fasta  --use-fasta-header
```

Updating a clustered database

It is possible to update previous clustered databases without re-clustering everything from scratch. Here is an example of how to proceed:

Let's create an older version of the DB.fasta by trimming out some sequences:

```
$ cd examples
$ awk '/^>/{seqCount++;} {if (seqCount <= 19000) {print $0;}}' DB.fasta > DB_trimmed.fasta
```

Now we create the sequence DBs, and cluster the old (trimmed) database:

```
$ mmseqs createdb DB.fasta DB_new
$ mmseqs createdb DB_trimmed.fasta DB_trimmed
$ mmseqs cluster DB_trimmed DB_trimmed_clu tmp
```

If you want to update DB_trimmed_clu with the newer version of your database DB_new

```
$ rm tmp/*
$ mmseqs clusterupdate DB_trimmed DB_new DB_trimmed_clu DB_clusterupdate tmp
```

DB_clusterupdate contains now the fresh clustering of DB_new.

Read more about updating cluster here.

Overview of Folders in MMseqs

- `bin`: `mmseqs`
- `data`: BLOSUM matrices and the workflow scripts (`blastp.sh`, `blastpgp.sh`, `cascaded_clustering.sh`, `linclust.sh`, `searchtargetprofile.sh`, `clustering.sh`)
- `examples`: test data `QUERY.fasta` and `DB.fasta`
- `util`: Contains the Bash parameter completion script.

Overview of MMseqs2 Commands

MMseqs2 contains five workflows that combine the core MMseqs2 modules (prefilter, align, kmermatcher, rescorediagonal and clust) and several other smaller ones.

Workflows:

- `mmseqs search`: Compares all sequences in the query database with all sequences in the target database, using the prefiltering and alignment modules. MMseqs2 search supports sequence/sequence, profile/sequence or sequence/profile searches.
- `mmseqs cluster`: Clusters sequences by similarity. It compares all sequences in the sequence DB with each other using mmseqs search, filters alignments according to user-specified criteria (max. E-value, min. coverage,...), and runs mmseqs clust to group similar sequences together into clusters.
- `mmseqs linclust`: Clusters sequences by similarity in linear time. It clusteres magnitudes faster than `mmseqs cluster` but is less sensitive.
- `mmseqs clusterupdate`: MMseqs2 incrementally updates a clustering, given an existing clustering of a sequence database and a new version of this sequence database (with new sequences being added and others having been deleted).
- `mmseqs taxonomy` Taxonomy assignment by computing the lowest common ancestor of homologs.

And the four core modules:

- `mmseqs prefilter`: Computes k-mer similarity scores between all sequences in the query database and all sequences in the target database.

- `mmseqs kmermatcher`: finds exact k-mer matches between all input sequences in linear time.
- `mmseqs align`: Computes Smith-Waterman alignment scores between all sequences in the query database and the sequences of the target database whose prefiltering scores computed by `mmseqs prefilter` pass a minimum threshold.
- `mmseqs clust`: Computes a similarity clustering of a sequence database based on Smith Waterman alignment scores of the sequence pairs computed by `mmseqs align`.

MMseqs2 has more than 30 modules in total. We provide modules for clustering, searching, alignments, taxonomy, and data transformation. For a complete list of all available modules, execute `mmseqs` without arguments.

Description of Workflows

Batch Sequence Searching using `mmseqs search`

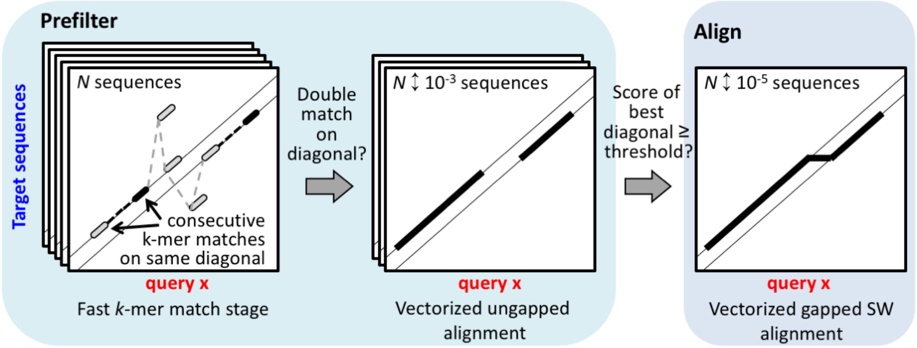
Prefilter
Target sequences
N sequences
consecutive k-mer matches on same diagonal
query x
Fast k-mer match stage
Double match on diagonal?
N ↕ 10^-3 sequences
query x
Vectorized ungapped alignment
Score of best diagonal ≥ threshold?
Align
N ↕ 10^-5 sequences
query x
Vectorized gapped SW alignment

Figure 1: Search workflow

For searching a database, query and target database have to be converted by `createdb` in order to use them in MMseqs. The search can be executed by typing:

```
$ mmseqs search queryDB targetDB outDB tmp
```

MMseqs2 supports iterative searches which are similar to PSI-BLAST. The following program call will run two iterations through the database. In the first iteration sequences are searched against sequence and in the second one profiles are used to search against sequences.

MMseqs2 will use the output for the first iteration sequence-sequence search to computes a profile (result2profile). The profile will be used as input in the next search iteration.

```
$ mmseqs search queryDB targetDB outDB tmp --num-iterations 2
```

This workflow combines the prefiltering and alignment modules into a fast and sensitive batch protein sequence search that compares all sequences in the query database with all sequences in the target database.

Query and target databases may be identical. The program outputs for each query sequence all database sequences satisfying the search criteria (such as sensitivity).

MMseqs2 can precompute the prefilter index `createindex` to speed up subsequence prefilter index read-ins. We recommend to use an index for iterative searches or if a target database will be reused several times. However reading the index can be bottle neck when using a network file systems (NFS). It is recommended to keep the index on a local hard drive. If storing the index file on a local hard drive is not possible and the NFS is a bottleneck than do not precompute the index. MMseqs2 will compute an index on the fly which reduces the IO volume by roughly a factor of seven.

The underlying algorithm is explained in more detail in section Computation of Prefiltering Scores using mmseqs prefilter, and the important parameter list can be found in section Search Workflow.

Translated Sequence Searching

The search workflow can handle nucleotide as query or target database. It will trigger a BLASTX or TBLASTN search respectively. The search detects the open reading frames on all six frames and translates them into proteins. As default the minimum codon length of 30 (10 amino acids) is used.

To perform a search a BLASTX or TBLASTN create your database by using `createdb`. It can automatically detect if the input are amino acids or nucleotides.

```
mmseqs createdb ecoli.fna ecoli_genome --dont-split-seq-by-len
mmseqs createdb ecoli.faa ecoli_proteins
```

A BLASTX like search can be triggered using the nucleotide database on the query database side.

```
mmseqs search ecoli_genome ecoli_proteins alnDB tmp
```

A TBLASTN like search can be triggered using the nucleotide database on the target database side.

```
mmseqs search ecoli_proteins ecoli_genome alnDB tmp
```

It is not possible to use nucleotide databases on query and target sides (TBLASTX) of the search workflow. The following workflow can be used to perform a TBLASTX search:

```
mmseqs extractorfs genome genome_orfs --min-length 30 --max-length 48000
mmseqs translatenucs genome_orfs genome_orfs_aa
mmseqs translatenucs ecoli_genome ecoli_genome_aa
mmseqs search genome_orfs_aa ecoli_genome alnDB tmp
mmseqs offsetalignment genome_orfs ecoli_genome_aa alnDB alnOffsetedDB
```

All open reading frames (ORFs) from each six frames can be extracted by using `extractorf`. This ORFs are translated into proteins by `translatenucleotide`. The tool `offsetalignment` will offset the alignment position to the `orf start position + alignment start * 3`.

Clustering Databases using `mmseqs cluster` or `mmseqs linclust`

To cluster a database, MMseqs2 needs a sequence database converted with `createdb` and an empty directory for temporary files. Then, you can run the cascaded clustering with:

```
$ mmseqs cluster inDB outDB tmp
```

The sensitivity of the clustering can be adjusted with the `-s` option. MMseqs2 will automatically adjust the sensitivity based on the `--min-seq-id` parameter, if `-s` is not provided.

Linclust can be used by calling `linclust`. The sensitivity can be adjusted by `--kmer-per-seq` (default 20).

```
$ mmseqs linclust inDB outDB tmp
```

The clustering workflow `cluster` combines the prefiltering, alignment and clustering modules into either a simple clustering or a cascaded clustering of a sequence database. There are two ways to execute the clustering:

- The *Simple clustering* `--single-step-clustering` runs the hashclust and prefiltering, alignment and clustering modules with predefined parameters with a single iteration.
- *Cascaded clustering* clusters the sequence database using the as first step linclust and then prefiltering, alignment and clustering modules incrementally in three steps.

Clustering criteria MMseqs2/Linclust and Linclust has three main criteria to link two sequences by an edge:

(1) a maximum E-value threshold (option `-e [0,∞[`) computed ac- cording to the gap-corrected Karlin-Altschul statistics using the ALP library;

(2) a minimum coverage (option `-c [0,1]`, which is defined by the number of aligned residue pairs divided by either the maximum of the length of query/centre and target/non-centre sequences `alnRes/max(qLen,tLen)` (default mode, `--cov-mode 0`), or by the length of the target/non-centre sequence `alnRes/tLen` (`--cov-mode 1`), or by the length of the query/centre `alnRes/qLen` (`--cov-mode 2`);

(3) a minimum sequence identity (`--min-seq-id` [0,1]) with option `--alignment-mode 3` defined as the number of identical aligned residues divided by the number of aligned columns including internal gap columns, or, by default, defined by a highly correlated measure, the equivalent similarity score of the local alignment (including gap penalties) divided by the maximum of the lengths of the two locally aligned sequence segments. The score per residue equivalent to a certain sequence identity is obtained by a linear regression using thousands of local alignments as training set.

Cascaded Clustering The cascaded clustering workflow first runs `linclust`, our linear-time clustering module, that can produce clustering's down to 50% sequence identity in very short time.

To achieve lower sequence identities and/or to further improve the resulting clusters, we continue with three cascaded clustering steps: In the first step of the cascaded clustering the prefiltering runs with a low sensitivity of 1 and a very high result significance threshold, in order to accelerate the calculation and search only for hits with a very high sequence identity. Then alignments are calculated and the database is clustered. The second step takes the representative sequences of the first clustering step and repeats the prefiltering, alignment and clustering steps. This time, the prefiltering is executed with a higher sensitivity and a lower result significance threshold for catching sequence pairs with lower sequence identity. In the last step, the whole process is repeated again with the final target sensitivity. At last, the clustering results are merged and the resulting clustering is written to the output database.

Cascaded clustering yields more sensitive results than simple clustering. Also, it allows very large cluster sizes in the end clustering resulting from cluster merging (note that cluster size can grow exponentially in the cascaded clustering workflow), which is not possible with the simple clustering workflow because of the limited maximum number of sequences passing the prefiltering and the alignment. Therefore, we strongly recommend to use cascaded clustering especially to cluster larger databases and to obtain maximum sensitivity.

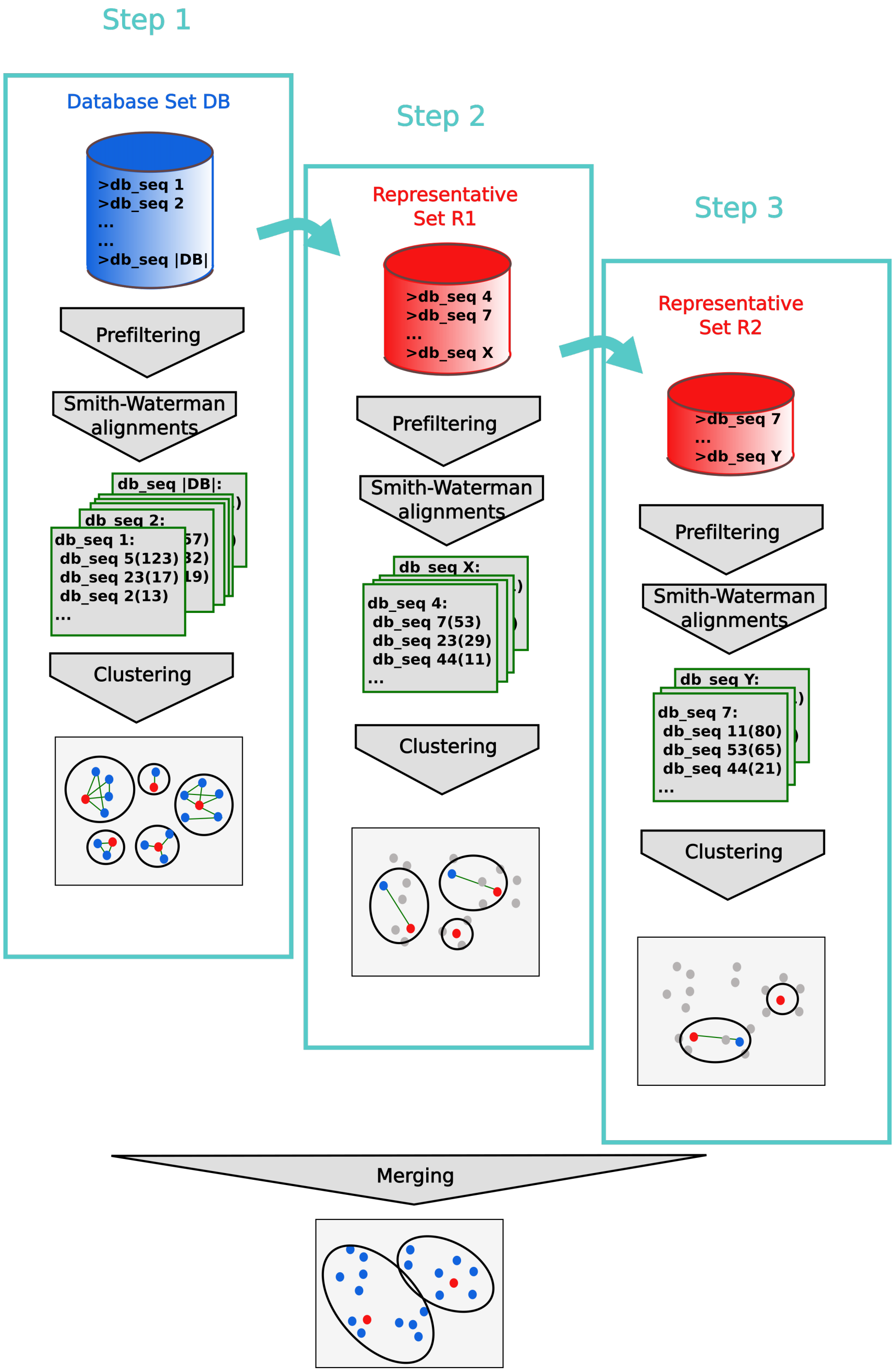

Figure 2: Cascaded clustering

Clustering modes All clustering modes transforms the alignment results into an undirected graph. In this graph notation, the verticies represents the proteins, which are connected by an edge. An edge between proteins are introduced if the alignment criteria (e.g. `--min-seq-id`, `-c` and `-e`) are fulfilled.

Greedy Set cover (`--cluster-mode 0`) algorithm is an approximation for the NP-complete optimization problem called set cover.

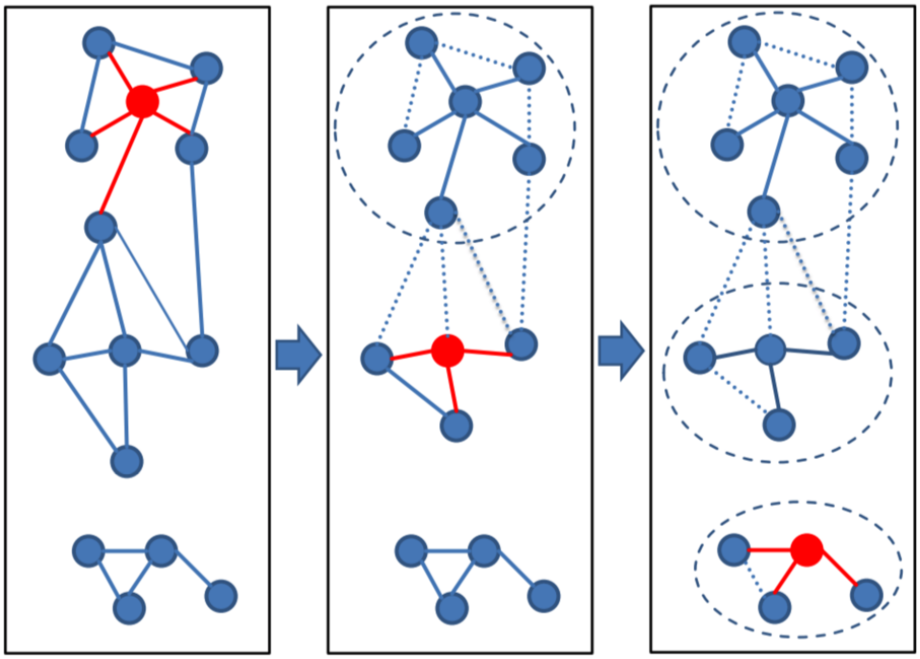

Figure 3: Set Cover clustering

Greedy set cover removes the node with most connections and all connected nodes. These forms a cluster and the procedure repeats until all nodes are in a cluster. The greedy set cover is followed by a reassignment step. Cluster member are assigned to another cluster centroid if their alignment score was higher.

Connected component (`--cluster-mode 1`) uses transitive connection to cover more remote homologs.

In connected component clustering starting at the mostly connected vertex, all vertices that are reachable in a breadth-first search are members of the cluster.

Greedy incremental (`--cluster-mode 2`) works analogous to CD-HIT clustering algorithm.

Greedy incremental clustering takes the longest sequence (indicated by the size of the node) and puts all connected sequences in that cluster, then repeatedly the longest sequence of the remaining set forms the next cluster.

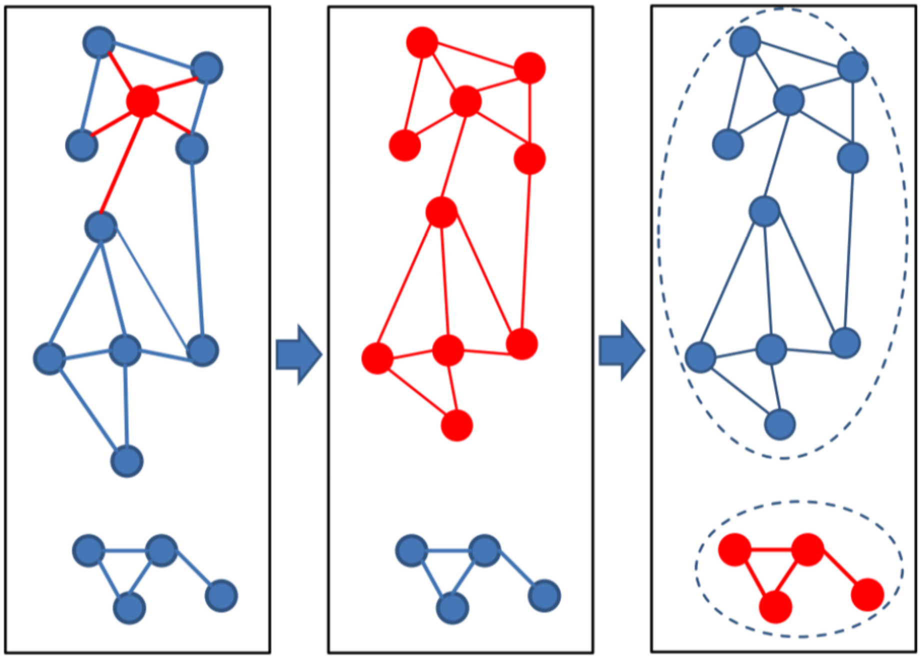

Figure 4: Connected component clustering

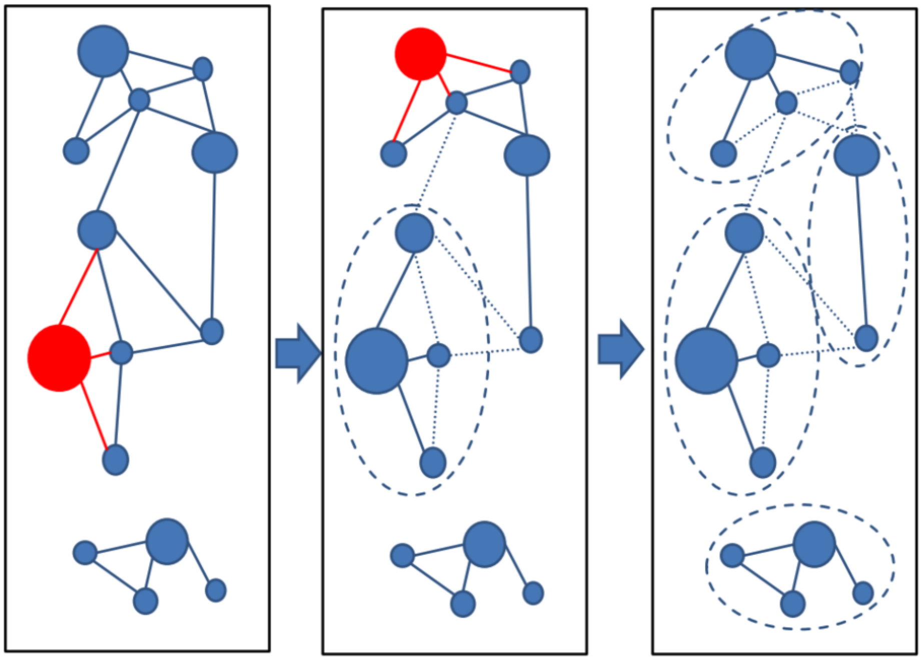

Figure 5: Greedy incremental clustering

Linear time clustering using `mmseqs linclust`

Linclust can cluster sequences down to 50% pairwise sequence similarity and its runtime scales linearly with the input set size.

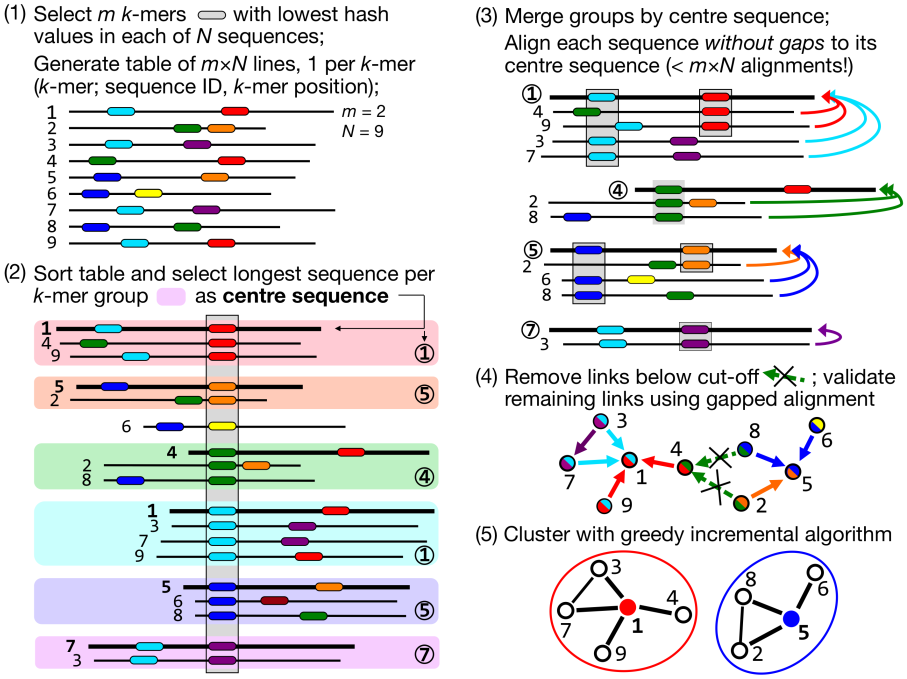
(1) Select m k-mers with lowest hash values in each of N sequences;
Generate table of m×N lines, 1 per k-mer (k-mer; sequence ID, k-mer position);
m = 2
N = 9
1
2
3
4
5
6
7
8
9
(2) Sort table and select longest sequence per k-mer group as centre sequence
①
⑤
④
⑦
(3) Merge groups by centre sequence;
Align each sequence without gaps to its centre sequence (< m×N alignments!)
(4) Remove links below cut-off ; validate remaining links using gapped alignment
(5) Cluster with greedy incremental algorithm

Figure 6: Linclust algorithm

Linear-time clustering algorithm. Steps 1 and 2 find exact k-mer matches between the N input sequences that are extended in step 3 and 4.

(1) Linclust selects in each sequence the `m` (default: 20) k-mers with the lowest hash function values, as this tends to select the same k-mers across homologous sequences. It uses a reduced alphabet of 13 letters for the k-mers and sets k=10 for sequence identity thresholds below 90% and k=14 above. It generates a table in which each of the mN lines consists of the *k*-mer, the sequence identifier, and the position of the *k*-mer in the sequence.

(2) Linclust sorts the table by *k*-mer in quasi-linear time, which identifies groups of sequences sharing the same k-mer (large shaded boxes). For each k-mer group, it selects the longest sequence as centre. It thereby tends to select the same sequences as centre among groups sharing sequences.

(3) It merges k-mer groups with the same centre sequence together: red + cyan and orange + blue and compares each group member to the centre

sequence in two steps: by global Hamming distance and by gapless local alignment extending the k-mer match.

(4) Sequences above a score cut-off in step 3 are aligned to their centre sequence using gapped local sequence alignment. Sequence pairs that satisfy the clustering criteria (e.g. on the E-value, sequence similarity, and sequence coverage) are linked by an edge.

(5) The greedy incremental algorithm finds a clustering such that each input sequence has an edge to its cluster's representative sequence. Note that the number of sequence pairs compared in steps 3 and 4 is less than mN, resulting in a linear time complexity.

Run Linclust Linclust needs a sequence database created by `createdb` and an empty directory for temporary files. Then, you can run the clustering with the following command:

```
$ mmseqs linclust inDB outDB tmp
```

Increasing the k-mers selected per sequence increases the sensitivity of linclust at a moderate loss of speed. Use the paramter `--kmer-per-seq` to set the number of k-mers selected per sequence. More k-mers per sequences results in a higher sensitivity.

The output format of linclust is the same format as in `mmseqs cluster`. See section Clustering Format.

Updating a Database Clustering using `mmseqs clusterupdate`

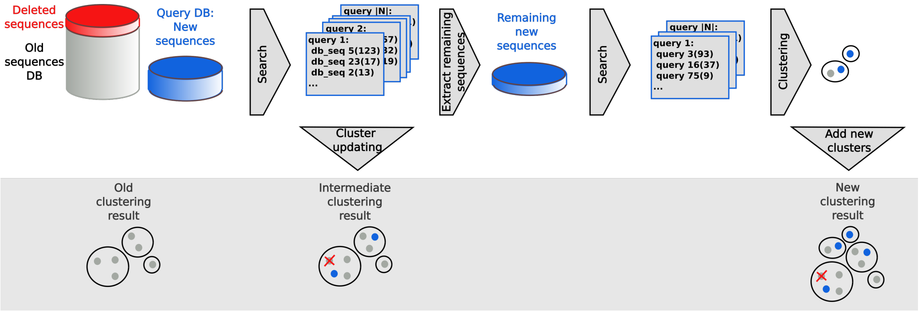

Figure 7: Update clustering

To run the updating, you need the old and the new version of your sequence database in sequence db format, the clustering of the old database version and a directory for the temporary files:

```
$ mmseqs clusterupdate oldDB newDB oldDB_clustering outDB tmp
```

This workflow efficiently updates the clustering of a database by adding new and removing outdated sequences. It takes as input the older sequence database, the results obtained by this older database clustering, and the newer version of the sequence database. Then it adds the new sequences to the clustering and removes the sequences that were removed from the newer database. Sequences which are not similar enough to any existing cluster will be representatives of new clusters.

Taxonomy assignment using `mmseqs taxonomy`

By identifying homologs through searches with taxonomy annotated reference databases, MMseqs2 can compute the lowest common ancestor. This lowest common ancestor is a robust taxonomic label for unknown sequences.

MMseqs2 implements the 2bLCA protocol (Hingamp et. al., 2013) with `--lca-mode 2` (default) for choosing a robust LCA.

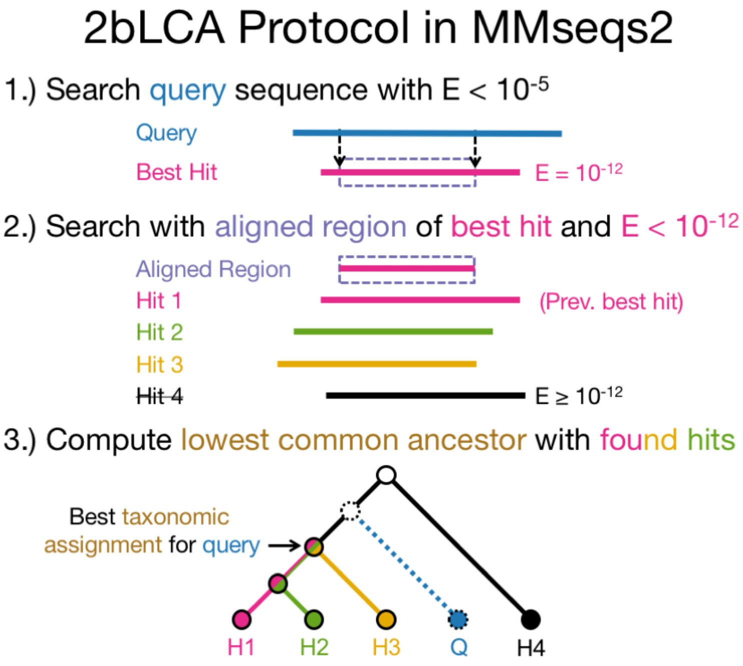
2bLCA Protocol in MMseqs2
1.) Search query sequence with E < 10^{-5}
Query
Best Hit
E = 10^{-12}
2.) Search with aligned region of best hit and E < 10^{-12}
Aligned Region
Hit 1
(Prev. best hit)
Hit 2
Hit 3
~~Hit 4~~
E ≥ 10^{-12}
3.) Compute lowest common ancestor with found hits
Best taxonomic assignment for query
H1
H2
H3
Q
H4

Figure 8: 2bLCA protcol (Hingamp et. al, 2013)

The second search can be disabled with `--lca-mode 1`. The LCA will then be only computed through the usual search workflow parameters (`--max-accept`, `-e`, etc.).

The LCA implementation is based on the Go implementation of blast2lca software on GitHub. It implements the LCA computation efficiently through *Range Minimum Queries* through an dynamic programming approach.

Prerequisites The taxonomy workflow requires the NCBI taxonomy taxdump.tar.gz. It is available on the NCBI FTP server:

```
mkdir ncbi-taxdump && cd ncbi-taxdump
wget ftp://ftp.ncbi.nih.gov/pub/taxonomy/taxdump.tar.gz
tar xzvf taxdump.tar.gz
cd ..
```

Provide the path to the extraction location in the `mmseqs taxonomy` call as the `<i:NcbiTaxdmpDir>` parameter.

The workflow further requires a tab-separated mapping `<i:targetTaxonMapping>` with every target database identifier mapped to a NCBI taxon identifier. The `convertkb` module can generate this mapping for any database with UniProt accessions, such as the Uniclust, UniRef, and the UniProt itself:

```
# Turn the target sequences into a MMseqs2 database (this also creates targetDB.lookup)
# Skip this step if you already created a database
mmseqs createdb target.fasta targetDB

# The targetDB.lookup file should be in the following format:
# numeric-db-id tab-character UniProt-Accession (e.g. Q6GZX4)

# UniRef has a prefixed accession (e.g. UniRef100_Q6GZX4)
# Remove this prefix first:
# sed -i 's|UniRef100_||g' targetDB.lookup

# Download the latest UniProt Knowledgebase:
wget ftp://ftp.expasy.org/databases/uniprot/current_release/knowledgebase/complete/uniprot_s
wget ftp://ftp.expasy.org/databases/uniprot/current_release/knowledgebase/complete/uniprot_t
cat uniprot_sprot.dat.gz uniprot_trembl.dat.gz > uniprot_sprot_trembl.dat.gz

# Generate annotation mapping DB (target DB IDs to NCBI taxa, line type OX)
mmseqs convertkb uniprot_sprot_trembl.dat.gz targetDB.mapping --kb-columns OX --mapping-file

# Reformat targetDB.mapping_OX DB into tsv file
mmseqs prefixid targetDB.mapping_OX targetDB.mapping_OX_pref
tr -d '\000' < targetDB.mapping_OX_pref > targetDB.tsv_tmp

# Cleanup: taxon format:  "NCBI_TaxID=418404 {ECO:0000313|EMBL:AHX25609.1};"
# Only the numerical identifier "418404" is required.
awk '{match($2, /=([^ ;]+)/, a); print $1"\t"a[1]; }' targetDB.tsv_tmp > targetDB.tsv
```

The `convertkb` module extracts either all or the chosen UniProt Knowledgebase line types into separate databases, which are indexed by their UniProt accession. By providing a tab separated mapping file between target database identifiers and UniProt accessions, a database of UniProt Knowledgebase entries, indexed by their target database identifiers, can be created. This database is then transformed into a tsv file.

Classification Once the prerequisites are generated, the taxonomy classification can be executed:

```
mmseqs taxonomy queryDB targetDB targetDB.tsv ncbi-taxdump queryLcaDB tmp
mmseqs createtsv queryDB queryLcaDB queryLca.tsv
```

Each line of the result file `queryLca.tsv` will contain a tab separated list of 1) query accession, 2) LCA NCBI taxon ID, 3) LCA rank name, and 4) LCA scientific name.

The `--lca-ranks` parameter can be supplied with a colon (`:`) separated string of taxonomic ranks. For example, `--lca-ranks genus:family:order:superkingdom` will resolve the respective ranks of the LCA and return a colon concatenated string of taxa as the fifth column of the result file.

Description of Core Modules

For advanced users, it is possible to skip the workflows and execute the core modules for maximum flexibility. Especially for the sequence search it can be useful to adjust the prefiltering and alignment parameters according to the needs of the user. The detailed parameter lists for the modules is provided in section Detailed Parameter List.

MMseqs2 contains three core modules: prefiltering, alignment and clustering.

Computation of Prefiltering Scores using `mmseqs prefilter`

The prefiltering module computes an ungapped alignment score for all consecutive k-mer matches between all query sequences and all database sequences and returns the highest score per sequence.

If you want to *cluster* a database, or do an all-against-all search, the same database will be used on both the query and target side. the following program call does an all-against-all prefiltering:

```
$ mmseqs prefilter sequenceDB sequenceDB resultDB_pref
```

sequenceDB is the base name of the mmseqs databases produced from the FASTA sequence databases by mmseqs createdb, the prefiltering results are stored in the mmseqs database files resultDB_pref andprefilterDB.index.

For *sequence search* two different input databases are usually used: a query database queryDB and a target database targetDB, though they can again be identical. In this case, the prefiltering program call is:

```
$ mmseqs prefilter queryDB targetDB resultDB_pref
```

MMseqs2 can handle profiles or protein sequences as input for the queryDB.

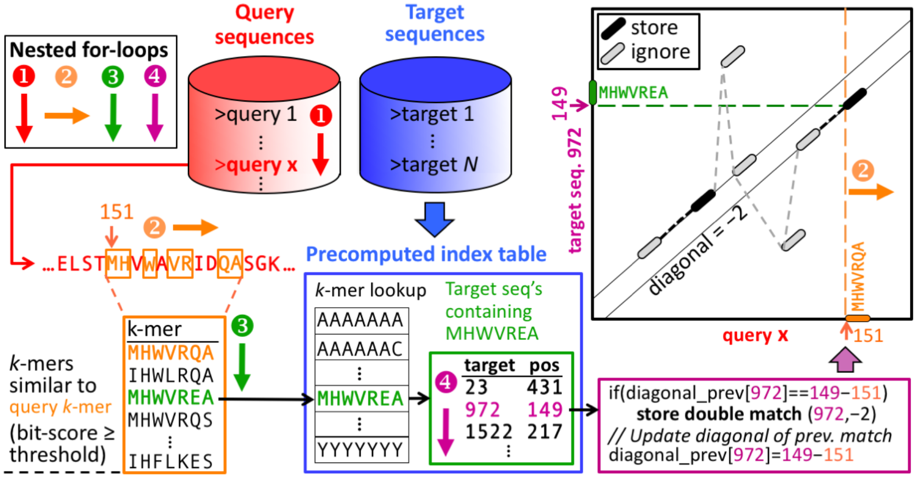

Figure 9: Prefilter

The prefilter k-mer match stage is key to the high speed and sensitivity. It detects consecutive short words ("k-mer") match on the same diagonal. The diagonal of a k-mer match is the difference between the positions of the two similar ''k"-mer in the query and in the target sequence.

The pre-computed index table for the target database (blue frame) contains for each possible ''k"-mer the list of the target sequences and positions where the k-mer occurs (green frame).

Query sequences/profiles are processed one by one (loop 1). For each overlapping, spaced query k-mer (loop 2), a list of all similar k-mer is generated (orange frame). The similarity threshold determines the list length and sets the trade-off between speed and sensitivity. The similiar k-mer list lenght can be controlled with -s.

For each similar k-mer (loop 3) we look up the list of sequences and positions where it occurs (green frame). In loop 4 we detect consecutive double matches on the same diagonals (magenta and black frames).

For each consecutive k-mer matches an ungapped alignment is computed. Only the maximal ungapped alignment score for each target is reported.

Set sensitivity `-s` parameter The sensitivity of the prefiltering can be set using the `-s` option. Internally, `-s` sets the average length of the lists of similar k-mers per query sequence position.

- *Similar k-mers list length*: Low sensitivity yields short similar k-mer lists. Therefore, the speed of the prefiltering increases, since only short k-mer lists have to be generated and less lookups in the index table are necessary. However, the sensitivity of the search decreases, since only very similar k-mers are generated and therefore, the prefiltering can not identify sequence pairs with low sequence identity.

It is possible to speed best hits searches by stepwise increasing `-s`. MMseqs2 includes a workflow for this purpose. How to find the best hit the fastest way

The following graphic shows the average AUC sensitivity versus speed-up factor relative to BLAST for 637,000 test searches. White numbers in plot symbols give number of search iterations.

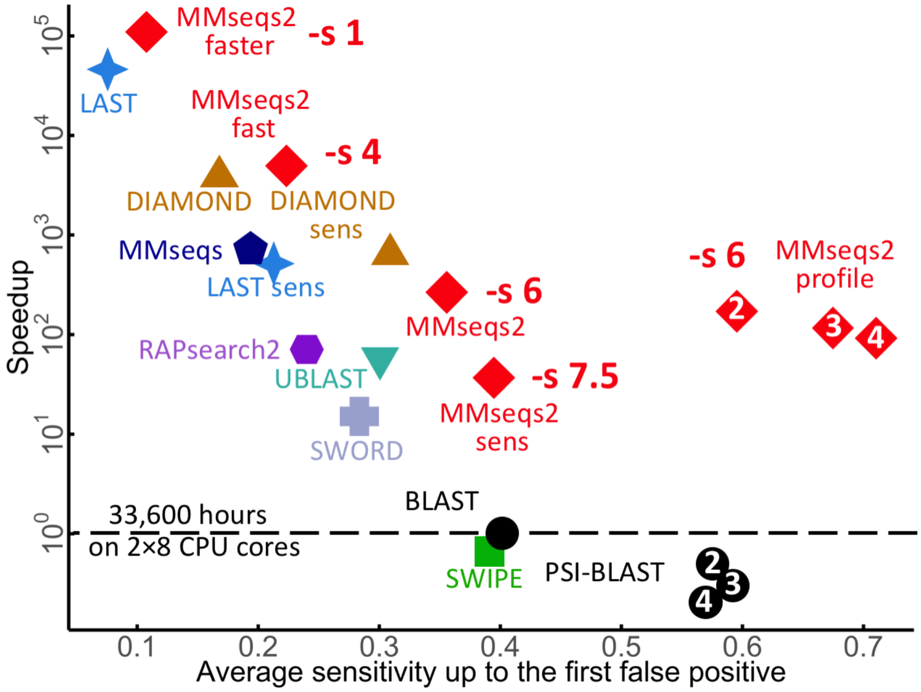

Figure 10: Prefilter sense

It is furthermore possible to use change the k-mer lengths, which are used in the prefiltering. Longer k-mers are more sensitive, since they cause less chance matches. Though longer k-mers only pay off for larger databases, since more time is needed for the k-mer list generation, but less time for database matching. Therefore, the database matching should take most of the computation time, which is only the case for large databases. As default MMseqs try to compute the optimal k-mer length based on the target database size.

Local alignment of prefiltering sequences using `mmseqs align`

In the alignment module, you can also specify either identical or different query and target databases. If you want to do a clustering in the next step, the query and target databases need to be identical:

```
$ mmseqs align sequenceDB sequenceDB resultDB_pref resultDB_aln
```

Alignment results are stored in the database files `resultDB_aln` and `resultDB_aln.index`.

Program call in case you want to do a sequence search and have different query and target databases:

```
$ mmseqs align queryDB targetDB resultDB_pref resultDB_aln
```

This module implements a SIMD accelerated Smith-Waterman-alignment (Farrar, 2007) of all sequences that pass a cut-off for the prefiltering score in the first module. It processes each sequence pair from the prefiltering results and aligns them in parallel, calculating one alignment per core at a single point of time. Additionally, the alignment calculation is vectorized using SIMD (single instruction multiple data) instructions. Eventually, the alignment module calculates alignment statistics such as sequence identity, alignment coverage and e-value of the alignment.

Clustering sequence database using `mmseqs clust`

For calling the stand-alone clustering, you need the input sequence database and a result database:

```
$ mmseqs cluster sequenceDB resultsDB_aln resultsDB_clu
```

Clustering results are stored in the MMseqs database files `resultsDB_clu` and `resultsDB_clu.index`.

The clustering module offers the possibility to run three different clustering algorithms by altering the `--cluster-mode` parameter. A greedy set cover algorithm is the default (`--cluster-mode 0`). It tries to cover the database by as few clusters as possible. At each step, it forms a cluster containing the representative sequence with the most alignments above the special or default thresholds with other sequences of the database and these matched sequences. Then, the sequences contained in the cluster are removed and the next representative sequence is chosen.

The second clustering algorithm is a greedy clustering algorithm (`--cluster-mode 2`), as used in CD-HIT. It sorts sequences by length and in each step forms a cluster containing the longest sequence and sequences that it matches. Then, these sequences are removed and the next cluster is chosen from the remaining sequences.

The third clustering algorithm is the connected component algorithm. This algorithm uses the transitivity of the relations to form larger clusters with more remote homologies. This algorithm adds all proteins to a cluster, that are reachable in a breadth first search starting at the representative with the most connections.

Output File Formats

MMseqs Database Format

Most MMseqs2 commands use the MMseqs database format (https://github.com/soedinglab/ffindex_soedinglab). The format is inspired by ffindex, which was developed by Andreas Hauser. It avoids drastically slowing down the file system when millions of files would need to be written or accessed, e.g. one file per query sequence in a many-to-many sequence search. MMseqs databases hide these files from the file system by storing them in a single data file. The *data file* `<name>` contains the data records, i.e. the contents of the file, concatenated and separated by \0 characters. A second, *index file* `<name>.index` contains for each numerical identifies (corresponding to the file name) the position of the corresponding data record in the data file.

Each line of the *index file* contains, separated by tabs, (1) the ID, (2) the offset in bytes of the data_record counted from the start of the data file, and (3) the size of the data record. The IDs have to be sorted numerically in ascending order, since for accessing a data record by IDs the matching IDs are found by binary search.

Here is an example for a database containing four sequences:

```
PSSLDIRL\0GTLKRLSAHYTPAW\0AEAIFIHEG\0YTHGAGFDNDI\0
```

The corresponding index file (file extension `.index`) could look like this.

```
10  0   9
11  9  15
12  24  10
13  34  12
```

The index contains four IDs, one for each data record: 10, 11, 12 and 13. The corresponding data records have offset positions 0, 9, 25, 35 and the data record sizes are 9, 15, 10, and 12 respectively.

The MMseqs2 modules `createdb` and `createfasta` do the format conversion from fasta to the MMseqs database format. `createdb` generates an MMseqs database from a FASTA sequence database. It assigns each sequence in the FASTA file sequentially a numerical id. Sequences that are longer than 32768 letters are split. `createfasta` converts an MMseqs database to a FASTA formatted text file: the sequence headers contain the DB identifiers preceded by `>`, and the sequence is extracted from the corresponding data record of the DB's data file.

However, for fast access in very large databases it is advisable to use the MMseqs database directly without converting it to FASTA format. We provide several tools at http://github.com/soedinglab/ffindex_soedinglab/ (query, build and apply function on each entry) to work with MMseqs databases. The binary `ffindex_get` can be used to directly access single records stored in an MMseqs database.

Prefiltering format

Each data record consists of the prefilter results for one query sequence. The ID is the database accession code, a numerical identifier (ID) for the query that was sequentially assigned by `createdb`.

Each line in a data record reports on one matched database sequence and has the following format (white space = \tab):

```
targetID  -log(E-value)  diagonal
```

where `targetID` is the database identifier of the matched sequence, `-log(E-value)` is the ungapped negative logarithmic E-value of the match, and `diagonal` is the diagonal i-j (i = position in query, j = position in db sequence) on which the match occurs.

Example of a database record for prefiltering:

```
2        71        0
3        35        0
5        -2        8
```

The first line describes a match with database sequence 2 on diagonal 0 with a -log(e-value) of 71 (e-value 1.46e-31).

Alignment format

Each data record consists of the alignment results for one query sequence. The ID of the queries was sequentially assigned by createdb.

Each line in a data record reports on match, i.e., one database sequence aligned to the query. It has the following format (white space = \tab)

```
targetID  alnScore  seqIdentity  eVal  qStart  qEnd  qLen  tStart  tEnd  tLen  [alnCigar]
```

Here, targetID is the database identifier of the matched sequence, alnScore is the bit score of the alignment in half bits, seqIdentity is the sequence identity [0:1], eVal is the e-value of the match, qStart is the start position of the alignment in the query, qEnd is the end position of the alignment in the query, tStart and tEnd are the start and end positions in the target (i.e. the database sequence), tLenis the target sequence length, the optionalalnCigarstring encodes the alignment in compressed format and is only included in the results if option-a‘ was used in mmseqs2 search. The numbers preceding the three letters M, I, and D give the number of match positions in a block aligned without gaps, the number of insertions and of deletions, respectively.

Example data record for alignment results:

```
2 705 1.000 8.771e-207 0  372 373 0  372 373 373M
5 367 0.595 3.319e-105 29 372 373 21 364 369 52M3I126M3D163M
3 347 0.565 2.722e-99  13 367 373 20 367 373 10M5I53M3I118M1D166M
```

The first line with targetID 2 is an identity match. The last sequence 3 has a Smith-Waterman alignment score of 347, the sequence identity 0.565 and the e-value 2.722e-99, the query start and end position is 13,367 of the total length 373, the target start and end position is 20,367 of the total length 373, the alignment string is 10M5I53M3I118M1D166M.

Clustering format

Internal cluster format Each data record consists of the IDs of the members of one cluster. The ID refers to the representative sequence of that cluster, (usually assigned by createdb).

Each line in a data record contains one ID of a cluster member. The first line of each data record contains the ID of the representative sequence of that cluster.

Here is an example of a cluster record with 3 cluster members:

```
2
5
3
```

The 2 is the ID of the representatives sequence while 5 and 3 are further cluster members.

Cluster TSV format The internal format can be converted to a flat tsv file:

```
$ mmseqs createtsv sequenceDB sequenceDB resultsDB_clu resultsDB_clu.tsv
```

The `resultsDB_clu.tsv` file follows the following format:

```
#cluster-representative     cluster-member
ID1 ID1
ID1 ID25
ID1 ID32
ID1 ID10
ID4 ID4
ID4 ID534
```

All members of the clustering are listed line by line. The first column always contains the representative sequence, the second contains the cluster member. For the example the cluster with the representative sequence ID1 contains four members it self and ID25, ID32, ID10. ID are parsed from the header from the input database (see id parsing from headers).

Cluster Fasta like format The internal format can be converted to a fasta a like format:

```
mmseqs createseqfiledb DB clu clu_seq
mmseqs result2flat DB DB clu_seq clu_seq.fasta
```

The resulting fasta a like format file will look like this:

```
>ID1
>ID1
MAGA....R
>ID25
MVGA....R
>ID32
MVGA....R
```

```
>ID10
MVGV....R
>ID4
>ID4
MCAT...Q
>ID534
MCAR...Q
```

A new cluster is marked by two identical name lines of the representative sequence, where the first line stands for the cluster and the second is the name line of the first cluster sequence. It is followed by the fasta formatted sequences of all its members.

Extract representative sequence To extract the representative of a clustering use the following commands:

```
mmseqs result2repseq DB clu clu_rep
mmseqs result2flat DB DB clu_rep clu_rep.fasta --use-fasta-header
```

The resulting fasta will contain all representative sequences:

```
>ID1
MAGA....R
>ID4
MCAT...Q
```

Identifier parsing

MMseqs2 parses identifier from the fasta header when transforming a result DB into a flat file by using e.g. `createtsv`, `convertalis`, . . .). We support following fasta header types:

```
    Uniclust,
    Swiss-Prot,
    Trembl,
    GenBank,
    NCBI Reference Sequence,
    Brookhaven Protein Data Bank,
    GenInfo Backbone Id,
    Local Sequence identifier,
    NBRF PIR,
    Protein Research Foundation,
    General database identifier,
    Patents,
    NCBI GI
```

If none of the header supported could be detected than we extract everything from header start (excluding >) until the first whitespace.

Optimizing Sensitivity and Consumption of Resources

This section discusses how to keep the run time, memory and disc space consumption of MMseqs2 at reasonable values, while obtaining results with the highest possible sensitivity. These considerations are relevant if the size of your database exceeds several millions of sequences and are most important if the database size is in the order of tens of millions of sequences.

Prefiltering module

The prefiltering module can use a lot of resources (memory consumption, total runtime and disc space), if the parameters are not set appropriately.

Memory Consumption For maximum efficiency of the prefiltering, the entire database should be held in RAM. The major part of memory is required for the k-mer index table of the database. For a database containing `N` sequences with an average length `L`, the memory consumption of the index lists is `(N * L * 7) byte`. Note that the memory consumption grows linearly with the size of the sequence database. In addition, the index table stores the pointer array and two auxiliary arrays with the memory consumption of `a^k*8` byte, where `a` is the size of the amino acid alphabet (default a=20, does not include the unknown amino acid X) and k is the k-mer size. The overall memory consumption of the index table is

```
M = (7 * N * L + 8 a^k) byte
```

Therefore, the UniProtKB database version of April 2014 containing 55 million sequences with an average length 350 needs about 71 GB of main memory.

MMseqs2 will automatically split the target database if the computer has not enough main memory.

Runtime The prefiltering module is the most time consuming step. It can scale from minutes in runtime to days by adjusting the sensitivity setting. Searching with 637000 protein sequences against 30 Mio Uniprot seqeunces took around 12 minutes on a 16 cores.

Disc Space The prefiltering results for very large databases can grow to considerable sizes (in the order of TB) of the disc space if very long result lists are allowed and no strict ungapped score threshold is set. As an example, an all-against-all prefiltering run on the 25 Mio seqeunces with `--max-seqs` 300 yielded prefiltering list with an average length of 150 and an output file size of 78 GB. One entry needs roughly 21 byte of space. To compute the worse case hard disk space usage `S` use the following formular. `N` is the Database sequence size L is `--max-seqs`.

```
S = (21 * N * L) byte
```

Important Options for Tuning the Memory, Runtime and Disc Space Usage

- The option `-s` controls the sensitivity in the MMseqs2 prefiltering module. The lower the sensitivity, the faster the prefiltering becomes, though at the cost of search sensitivity. See Set sensitivity -s parameter.
- The option `--max-seqs`controls the maximum number of prefiltering results per query sequence. For very large databases (tens of millions of sequences), it is a good advice to keep this number at reasonable values (i.e. the default value 300). For considerably larger values of `--max-seqs`, the size of the output can be in the range of several TB of disc space for databases containing tens of millions of sequences. Changing `--max-seqs` option has no effect on the run time but can degrade the sensitivity.

Alignment Module

In the alignment module, generally only the total runtime and disk space are the critical issues.

Memory Consumption The major part of the memory is required for the three dynamic programming matrices, once per core. Since most sequences are quite short, the memory requirements of the alignment module for a typical database are in the order of a few GB.

Runtime The alignment is based on a striped vectorized algorithm which can process roughly 2 giga cell updates per second (GCUPS). The time to compute the alignment of two average sized proteins (350 residues) takes roughly 6.0625E-5 seconds on one CPU. For example computing 23 Mio. alignments on 8 cores takes 2 minutes.

If a huge amount of alignments have to be calculated, the run time of the alignment module can become a bottleneck. The run time of the alignment module depends essentially on two parameters:

- The option `--max-seqs` controls the maximum number of sequences aligned with a query sequence. By setting this parameter to a lower value, you accelerate the program, but you may also lose some meaningful results. Since the prefiltering results are always ordered by their significance, the most significant prefiltering results are always aligned first in the alignment module.
- The option `--max-accept` controls the maximum number of alignment results per query sequence.
- The option `--max-rejected` defines the maximum number of rejected sequences for a query until the calculation of alignments stops. A reject is an alignment whose statistics don't satisfy the search criteria such as coverage threshold, e-value threshold etc. Per default, `--max-rejected` is set to `INT_MAX`, i.e. all alignments until `--max-seqs` alignments are calculated.

Disc Space Since the alignment module takes the results of the prefiltering module as input, the size of the prefiltering module output is the point of reference. If alignments are calculated and written for all the prefiltering results, the disc space consumption is 1.75 times higher than the prefiltering output size.

Clustering Module

In the clustering module, only the memory consumption is a critical issue.

Memory Consumption The clustering module can need large amounts of memory. The memory consumption for a database containing `N` sequences and an average of `r` alignment results per sequence can be estimated as

```
M = (6 * N * r) byte
```

To prevent excessive memory usage for the clustering of large databases, MMseqs2 uses a cascaded clustering by default, which accumulates sequences per cluster incrementally.

If you run the clustering module separately, you can tune the following parameters:

- `--max-seqs` parameter which controls the maximum number of alignment results per query considered (i.e. the number of edges per node in the graph). Lower value causes lower memory usage and faster run times.
- Alternatively, `-s` parameter can be set to a higher value in order to cluster the database down to higher sequence identities. Only the alignment results above the sequence identity threshold are imported and it results in lower memory usage.

Runtime Clustering is the fastest step. It needs less than an hour for the clustering of the whole UniProtKB.

Disc Space Since only one record is written per cluster, the memory usage is a small fraction of the memory usage in the prefiltering and alignment modules.

Workflows

The search and clustering workflows offer the possibility to set the sensitivity option `-s` and the maximum sequences per query option `--max-seqs`. `--max-rejected` option is set to `INT_MAX` per default. Cascaded clustering sets all the options controlling the size of the output, speed and memory consumption, internally adjusting parameters in each cascaded clustering step.

How to run MMseqs2 on multiple servers using MPI

MMseqs2 can run on multiple cores and servers using OpenMP (OMP) and message passing interface (MPI). MPI assigns database splits to each servers and each server computes them using multiple cores (OMP). Currently `prefilter`, `align`, `result2profile`, `swapresults` can take advantage of MPI. To parallelize the time-consuming k-mer matching and gapless alignment stages prefilter among multiple servers, two different modes are available. In the first, MMseqs2 can split the target sequence set into approximately equal-sized chunks, and each server searches all queries against its chunk. Alternatively, the query sequence set is split into equal-sized chunks and each server searches its query chunk against the entire target set. The number of chunks is controlled through the `--split` parameter. Splitting the target database is less time-efficient due to the slow, IO-limited merging of results, but it reduces the memory required on each server to:

```
((7 * N * L) / #chunks + 21^k * 8) byte
```

Thus, it allows users to search through huge databases on servers with moderate memory sizes. If the number of chunks is larger than the number of servers, chunks will be distributed among servers and processed sequentially. By default, MMseqs2 automatically decides which mode to pick based on the available memory (assume that all machines have the same amount of memory). Make sure that MMseqs2 was compiled with MPI by using the `HAVE_MPI=1` flag (`cmake -DHAVE_MPI=1 -DCMAKE_BUILD_TYPE=Release -DCMAKE_INSTALL_PREFIX=. ..`). Our precomplied static version of MMseqs2 can not use MPI. To search with multiple server just call the search and add the RUNNER variable. The TMP folder has to be shared between all nodes (e.g. NFS)

```
RUNNER="mpirun -np 42" mmseqs search queryDB targetDB resultDB tmp
```

For clustering just call the clustering. The TMP folder has to be shared between all nodes (e.g. NFS)

```
RUNNER="mpirun -np 42" mmseqs cluster DB clu tmp
```

Frequently Asked Questions

This section describes common questions.

How to set the right alignment coverage to cluster

MMseqs has three modes to control the sequence length overlap “coverage”: (1) bidirectional, (2) target coverage and (3) query coverage. In the context of `cluster` or `linclust`, the query is seen representative sequence and target is a member sequence. The `--cov-mode` flag also automatically sets the `--cluster-mode`.

(1) With `--cov-mode 0 -c [0.0,1.0]` only sequences are clustered that have a sequence length overlap greater than X% of the longer of the two sequences. This coverage mode should be used to cluster full length protein sequences. The multi domain structure of proteins will be most likely preserved when using a coverage > 80% (`-c 0.8`). Default `--cluster-mode` is the greedy set cover.

For example:

```
q: MAVGTACRPA
t: -AVGTAC---
```

The coverage of would be 6/10=60%

```
q: -AVGTAC---
t: MAVGTACRPA
```

The coverage of would be 6/10=60%

(2) With `--cov-mode 1 -c [0.0,1.0]` (target-cov mode) only sequences are clustered that have a sequence length overlap greater than X% of the target sequence. The target cov mode can be used to cluster protein fragments. To suppress fragments from becoming representative sequences, it is recommended to use `--cluster-mode 2` in conjunction with `--cov-mode 1`. Default `--cluster-mode` is the greedy incremental clustering (by length).

For example:

```
q: MAVGTACRPA
t: -AVGTAC---
```

The target coverage would be 6/6=100%

```
q: -AVGTAC---
t: MAVGTACRPA
```

The target coverage would be 6/10=60%

(3) With `--cov-mode 2 -c [0.0,1.0]` (query-cov mode) only sequences are clustered that have a sequence length overlap greater than X% of the query sequence. The query cov mode can be used while searching e.g. to assure a certain level of coverage.

For example:

```
q: MAVGTACRPA
t: -AVGTAC---
```

The query coverage would be 6/10=60%

```
q: -AVGTAC---
t: MAVGTACRPA
```

The query coverage would be 6/6=100%

How is MMseqs computing the sequence identity

There are two ways MMseqs can compute the sequence identity, both of which produce similar results:

(1) When using `--alignment-mode 3` mmseqs2 will compute the number of identical aligned residues divided by the number of aligned columns including columns containing a gap in either sequence.

(2) By default, the sequence identity is estimated from the score per column, i.e., the local alignment bit score divided by the maximum length of the two aligned sequence segments. The estimate uses the linear regression function (shown in red below) between the sequence identity computed as in (1) and the score per column in the scatter plot:

The score per column is a better measure of the degree of similarity than the actual sequence identity, because it also takes the degree of similarity between aligned amino acids and the number and length of gaps into account.

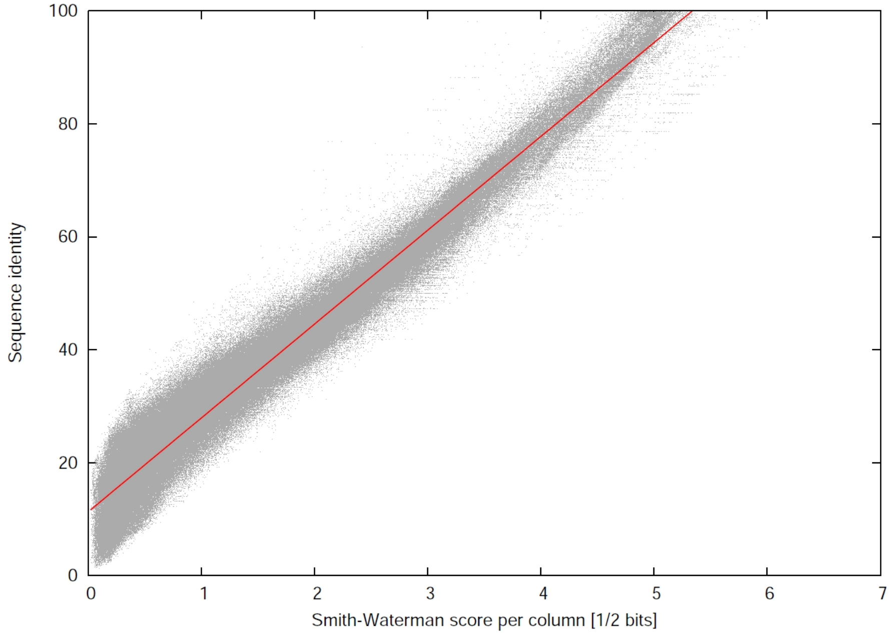

Figure 11: Relationship between score per column and sequence identity

How to restart a search or clustering workflow

MMseqs checks if files are already computed in the `tmpDir` and skips already computed results. To restart delete temporary result files from the crashing step that were created by MMseqs and restart the workflow with the same program call again. You can recognise the temporary files that should be deleted by their file ending `.[0-9]+`.

If the job crashed while merging files they can be merged manually using ffindex_build (https://github.com/soedinglab/ffindex_soedinglab). For example, if the merge step of the alignment fails while using 56 threads then the result could be recovered by using the following command.

```
for i in $(seq 0 55); do ffindex_build -a aln{,.tmp.index} -d aln.$i  -i aln.index.$i ; done
LC_ALL=C sort --parallel 28 -n -k 1,1 aln.tmp.index > aln.index
```

How to find the best hit the fastest way

MMseqs2 can apply an iterative approach to speed up best-hit-searches. It will start searching with the lowest sensitivity defined with `--start-sens` and search until the target sensitivity `-s` is reached. The amount of steps to reach `-s` can be defined with `--sens-steps`.

Queries are only used again in the next iteration, if no match could be found that fulfilled the acceptance criteria in the previous iteration.

For example, the following search performs three search steps with sensitivity `-s` 1, 4 and 7.

```
mmseqs search qDB tDB rDB tmp --start-sens 1 --sens-steps 3 -s 7
```

Using this iterative approach can speed up best-hit-searches 4-10 times.

There is a chance that the best hit is not found but the chances are low. Prefilter hits found at a lower sensitivity threshold, have more highly conserved k-mers in common. This effect can be reduced if a higher start sensitivity is used (–start-sens 4).

If any hit is good enough add the '–max-accept 1' option to gain a further speedup.

How is MMseqs handling low complexity

MMseqs uses reduces low complexity effects on the query and target database.

Query sequences are handled by an amino acid local compositional bias correction. In prefilter and alignment stages we apply a correction to substitution matrix scores assigning lower scores to the matches of amino acids that are overrepresented in the local sequence neighborhood. To switch the compositional bias correction on and off use `--comp-bias-corr`.

Target sequences low-complexity regions are masked during the prefilter stage. We use TANTAN with a threshold of 90% probability for low complexity. Masking can be controlled with `--mask`.

How to redundancy filter sequences with identical length and 100% length overlap.

To redundancy filter sequences of identical length and 100% overlap `mmseqs clusthash` can be used. It reduces each sequence to a five-letter alphabet, computes a 64 bit CRC32 hash value for the full-length sequences, and places sequences with identical hash code that satisfy the sequence identity threshold into the same cluster.

Example: cluster sequences at 90% sequence identity

```
mmseqs clusthash sequenceDB resultDB --min-seq-id 0.9
mmseqs clust sequenceDB resultDB clusterDB
```

How to add sequence identities and other alignment information to a clustering result.

We can add sequence identities and other alignment information to the clustering result `outDB` by running an additional align step:

```
$ mmseqs cluster sequenceDB resultDb tmp
$ mmseqs align sequenceDB sequenceDB resultDb alignDB -a
$ mmseqs convertalis sequenceDB sequenceDB alignDB align.m8
```

The `-a` parameter computes the whole backtrace. `--alignment-mode 3` could be used instead if the backtrace is not needed. This would save disk space. The backtrace is however computed anyway (for the calculation of the sequence identities) and then discarded.

How to run external tools for each database entry

The`apply` module can be used to call an external tool for each entry of a MMseqs2 database. It works like the map step from the map/reduce pattern. It calls for every index entry the specified process with the passed parameters. The process reads the entry data from stdin and its stdout is written to a new entry in the result database. The tool supports OpenMP and MPI parallelization for spreading out the job over several compute nodes.

Example: An awk script which takes an alignment result entry from stdin and prints out all lines with an e-value <0.001 to stdout (Hint: the `filterdb` module can also solve this problem, but with less overhead):

```
mmseqs apply resultDB filteredResultDB -- awk '$4 < 0.001 { print; }'
```

The `apply` module exports the `MMSEQS_ENTRY_NAME` environment variable into the called processes. It contains the current database key.

How to compute a multiple alignment for each cluster

There are two ways to produce MSAs from a clustering

(1) MMseqs2 `mmseqs result2msa` can produce a MSA using a centre star alignment without insertions in the query.

mmseqs cluster DB DB_clu tmp mmseqs result2msa DB DB DB_clu DB_clu_msa

(2) It is possible to call an external multiple aligner by using `mmseqs apply` tool. The multiple aligner need the capability to read stdin and write the result to stdout.

mmseqs cluster DB DB_clu tmp mmseqs createseqfiledb DB DB_clu DB_clu_seq mmseqs apply DB_clu_seq DB_clu_seq_msa – clustalo -i - –threads=1

How to manually cascade cluster

It is possible to cluster the representative sequences of an clustering run and merge the cluDB results with the following workflow.

```
# first clustering run
mmseqs linclust sequenceDB clu1 tmp1
# create a subset of the sequenceDB only with representative sequences
mmseqs createsubdb clu1 sequenceDB cluSequenceDB
# cluster representative sequences
mmseqs cluster cluSequenceDB clu2 tmp2
# merge two clusterings in to one results
mmseqs mergecluster sequenceDB final_clu clu1 clu2
```

How to cluster using profiles

The following workflow is a profile consensus clustering.

1.) Enrich the sequences:

```
# enrich your database to cluster (seqDB1) by searching it against a database seqDb2
mmseqs search seqDB1 seqDB2 resultDB1 tmp
# turn seqDB1 into profiles
mmseqs result2profile seqDB1 seqDB2 resultDB1 profileDB1
```

2.) Cluster profiles by searching the profiles against its consensus sequences

```
# search with profiles against consensus sequences of seqDB1
mmseqs search profileDB1 profileDB1_consensus resultDB2 tmp --add-self-matches  # Add your c
# cluster the results
mmseqs clust profileDB1 resultDB2 profileDB1_clu`
```

How to create a HHblits database

One can turn the output of a search (or clustering) into a HHblits database. You need to have HH-suite properly installed with MPI support. The following procedure creates an HHblits-compatible database “searchMsa” resulting from the enrichment of sequences of “DBquery” with the sequences of “DBtarget”:

```
mmseqs search DBquery DBtarget searchOut tmp -a
mmseqs result2msa DBquery DBtarget searchOut searchMsa --compress
mpirun -np 2 cstranslate_mpi -i searchMsa -o searchMsa_cs219 -A /path/to/cs219.lib -D /path/
```

The files `/path/to/cs219.lib` and `/path/to/context_data.lib` are provided in the "data" subfolder of your HH-suite installation. The parameters `-x 0.3 -c 4` have been empirically found to perform well.

For creating an HHblits database from a clustering, the procedure is almost the same, except that you have to create symlinks to the ffindex *header and* sequence files needed by HHblits:

```
mmseqs cluster DB clu tmp
mmseqs result2msa DB DB clu cluMsa --compress
ln -s DB_h cluMsa_header.ffdata
ln -s DB_h.index cluMsa_header.ffindex
ln -s DB cluMsa_sequence.ffdata
ln -s DB.index cluMsa_sequence.ffindex
mpirun -np 2 cstranslate_mpi -i cluMsa -o cluMsa_cs219 -A /path/to/cs219.lib -D /path/to/con
```

In the "search" case, those files are generated by MMseqs2, since it needs to merge the query and the target sequence databases. No merging is done for clustering, since both the query and target sequence database are the same.

How to create a target profile database (from PFAM)

Download the latest version of the PFAM in stockholm format:

```
wget http://ftp.ebi.ac.uk/pub/databases/Pfam/current_release/Pfam-A.full.gz
```

Convert stockholm MSAs to a FASTA formated MSA database. The pfam documentation says that the sequences in the pfam will not wrap around, thus this example awk script does not wrapping stockholm entries. You can adjust the `-v db=pfam_msa` parameter for a different output database name:

```
zcat Pfam-A.full.gz | tr '[a-z]' '[A-Z]' | awk -v db=pfam_msa \
    'BEGIN { idx = 1; offset = 0; } /^#=GF/ && $2 == "AC" { key=$3; next; } /^#/ { next; } /
```

Create a profile database in MMseqs2 format. Since pfam has no representative sequence use gap-rule for match mode, that will filter out columns with more than 50% gaps (of weighted sequences):

```
mmseqs msa2profile pfam_msa pfam_profile --match-mode 1
```

Precompute mmseqs index table (not required for a single search run). Use the `--no-preload` flag later in the search, if the query database is small to medium sized. Without that the precomputed index table will be first read completely into memory (unnecessary overhead).

```
mmseqs createindex pfam_profile tmp -k 5 -s 7
```

Search now against the created profile database:

```
mmseqs search query_db pfam_profile_new result tmp -k 5 -s 7
```

Workflow Control Parameters

Search Workflow

Compares all sequences in the query database with all sequences in the target database.

Usage:

```
mmseqs search <queryDB> <targetDB> <outDB> <tmpDir> [opts]
```

Options:

```
-s [float] Target sensitivity in the range [1:8.5] (default=4).
```

Adjusts the sensitivity of the prefiltering and influences the prefiltering run time. 1.0 fastest - 8.5 sensitive. The sensitivty between 8 to 8.5 should be as sensitive as BLAST. For detailed explanation see section Computation of Prefiltering Scores using mmseqs prefilter.

Clustering Workflow

Calculates the clustering of the sequences in the input database.

Usage:

```
mmseqs cluster <sequenceDB> <outDB> <tmpDir> [opts]
```

Options:

```
--single-step-clustering Starts the single-step instead of the
cascaded clustering workflow.
```

The database can be clustered in a single step instead of with a cascaded workflow. This increases runtime and memory requirements substantially and decreases sensitivity, but guarantees, that all cluster members strictly fulfill the selection criteria, such as sequence identity or coverage. After merging of clusters in the cascaded clustering, the e.g. sequence identity of the representative with the members of the to be merged cluster, might fall under the given sequence identity threshold.

`-s [float] Target sensitivity in the range [2:9] (default=4).`

Adjusts the sensitivity of the prefiltering and influences the prefiltering run time. For detailed explanation see section Computation of Prefiltering Scores using mmseqs prefilter.

`--min-seq-id [float] list matches above this sequence identity [0.0:1.0] (default=0.0).` Read more about how MMseqs is computing sequence identity in section How is MMseqs computing the sequence identity.

`--cov-mode [int] "0: coverage of query and target, 1: coverage of target [0:1] (default=0). -c [float] "list matches above this fraction of covered residues (see cov-mode) [0.0:1.0] (default=0.8).` Read more about coverage is computed at section How to set the right alignment coverage to cluster

Updating Workflow

Updates the existing clustering of the previous database version with new sequences from the current version of the same database.

Usage:

```
mmseqs clusterupdate <oldDB> <newDB> <oldDB_clustering> <outDB>
<tmpDir> [opts]
```

Options:

`--sub-mat [file] Amino acid substitution matrix file.`

Substitution matrices for different sequence diversities in the required format can be found in the MMseqs2 `data` folder.

External Libraries used in MMseqs2

We would also like to thank the developers of the open source librarys used in MMseqs2:

- Striped Smith-Waterman Library
- ALP Library
- TANTAN
- Open MP Template Library
- kseq
- iota
- blast2lca

Developers Guide

Regression test

To run a search regression test execute the following steps:

```
# download the runner script and set permissions
$ wget https://bitbucket.org/martin_steinegger/mmseqs-benchmark/raw/master/scripts/run_codes
$ chmod +x run_codeship_pipeline.sh

# change three varialbes in this file edit the following variables:
# If you dont have AVX2 on the machine just comment all lines containing MMSEQSAVX

BASE_DIR="$HOME/clone/regression_test" MMSEQSSSE="$HOME/clone/build/src/mmseqs"
MMSEQSAVX="$HOME/clone/build_avx2/src/mmseqs"

# run script and set CI_COMMIT_ID to some non-empty string (in our CI system this is automat
$ CI_COMMIT_ID="TESTING" ./run_codeship_pipeline.sh

# The script will return an error code != 0 if there is a regression in sensitivity of MMsec
$ [ $? -eq 1 ] && echo "Error"
```

It will print a report with sensitivity AUCs it achieved and then error out if it did not achieve the minimum AUCs. Currently 0.235 for normal sequence searches and 0.331 for profile searches.

You can also use our Docker images to run this benchmark:

```
cd mmseqs-folder
docker build -t mmseqs2 .
git clone https://bitbucket.org/martin_steinegger/mmseqs-benchmark.git
cd mmseqs-benchmark
docker build -t mmseqs-benchmark .
```

The regression test passed, if the second image exits cleanly.

Sanitizers

MMseqs2 can be built with ASan/MSan/UBSan/TSan support by specifying calling:

```
cmake -DHAVE_SANITIZER=1 -DCMAKE_BUILD_TYPE=ASan ...
```

Replace ASan with MSan, UBsan or TSan for the other sanitizers. CMake will error and abort if your compiler does not support the respective sanitizer.

License Terms

The software is made available under the terms of the GNU General Public License v3.0. Its contributors assume no responsibility for errors or omissions in the software.