FUZZY MATCHING

FIZZY FETCHING 3

Software Solution by Abdulaziz, Claudia, Gary, Joseph, Remy, and Zach

WHAT IS FUZZY MATCHING?

Fuzzy Matching (Approximate string matching) is the process of finding sequences that match a pattern approximately, not exactly.

- Where did it start?
  - Many-valued logic proposed by Jan Łukasiewicz in 1920
  - Evolution of Translation Memory tools leading up to the 90’s
- What’s it used for?
  - Traditionally for spell checking
  - Computer Assisted Translation
  - Approximate string matching on large DNA sequences

MOTIVATION

- *Proposed idea*
- *Connects to topics covered in course material*
  - *Alignment and matching concepts*
- *Flexibility of doing our own thing*
  - *Adding additional value to our product*

SOLUTION OVERVIEW

C++ command-line interface

Input

./run <FASTA file> <Substring>

./run E-coli.fna ACTGACTAGCGACTACGGCAG

Output

<Sequence>: <best substring> | <levenshtein> | <sorensen>

...

NC_000913.3 Escherichia coli str. K-12 substr. MG1655, complete genome: GAGTTTGATCAT | Levenshtein: 1 | Sorensen: 0.909091

...

MAIN LOGIC

1. **Parse** command-line arguments
2. **Read** in FASTA file
3. **Extract** k-mers from sequence
4. **Compare** each k-mer to the given substring (parallelized)
5. **Store** the best substring of each sequence
6. **Sort** rankings
7. **Output** smallest distances, in order

CODE OVERVIEW

Driver

- Glue code and CLI

Rank

- Provides ranking functionality w/ comparison operators
- **<Sequence ID> | <Substring match> | Score**

Sequence

- K-mer generation
- **<Sequence ID> | <Full sequence>**

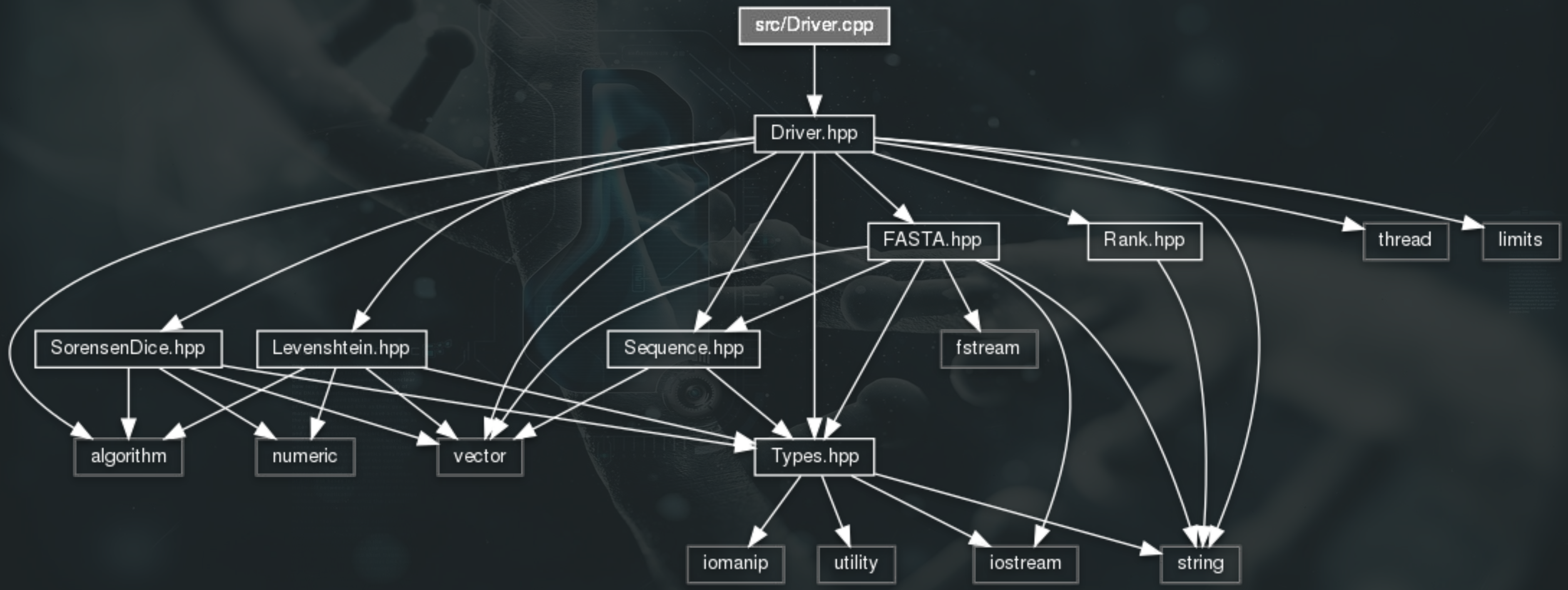
Distance metrics

- Levenshtein
- Sørensen–Dice

FASTA

- Reads a FASTA file and converts it into a vector of sequences

DEPENDENCY GRAPH



```cpp
// If we can assign a sequence to each thread, parallelize at the sequence-level.
// Otherwise, parallelize individual sequences. There is a greater overhead of
// parallelizing individual sequences because the CPU threads get re-assinged
// very quickly.
const unsigned concurrentThreadsSupported = std::thread::hardware_concurrency();
const bool parallelizeKMers = (sequences.size() < concurrentThreadsSupported) ? true : false;

// How many k-mers can we fit into the max memory amount?
// Non-const because it is modified on the last iteration
Index numKmers = maxMemory / substringLength;

// Go through each sequence
Index i;
#pragma omp parallel for schedule(dynamic) private(i) \
if (!parallelizeKMers && ENABLE_PARALLELIZATION)
for (i = 0; i < sequences.size(); ++i) {
    // Calculate how many k-mers to calculate at a time
    // For now get all substrings at once
    const Index numIters = (sequences.at(i).DNA.length() - 2*substringLength) / numKmers;
    // Prepare to store ranks
    std::vector<Rank> localRanks(numKmers);
    // Only generate a certain amount at a time
    const std::string id = sequences.at(i).id;
    for (Index j = 0; j <= numIters; ++j) {
        const auto substrings = \
        GetSubstrings(j * numKmers, substringLength, numKmers, sequences.at(i).DNA);

        // Now calculate the Levenshtein distance for each k-mer, parallelized
        #pragma omp parallel for schedule(dynamic) \
        private(j) firstprivate(id) \
        if (parallelizeKMers && ENABLE_PARALLELIZATION)
        for (Index k = 0; k < numKmers; ++k) {
            // Calculate distance
            localRanks.at(k) = \
            Rank(id,
                 substrings.at(k),
                 LevenshteinDistance(substrings.at(k), substring),
                 SorensenDice(substrings.at(k), substring));
        }
    }

    // Save best substring + score
    globalRanks.at(i) = *std::min_element(localRanks.begin(), localRanks.end());
}
```

ALGORITHMS

Levenshtein

- Calculates edit distance
- Integer result
- Simple

Ex: Kitten → Sitting

1. Kitten → Sitten (Score += 1)
2. Sitten → Sittin (Score += 1)
3. Sittin → Sitting (Score += 1)

Total Levenshtein Distance of 3.

Sørensen–Dice

- Calculates similarity percentage
- Decimal result between 0 & 1
- Mostly simple

Ex: Night v.s Nacht

1. Get set of bigrams for each word:
   a. A = {Ni, ig, gh, ht}
   b. B = {Na, ac, ch, ht}
2. Get length of each set. (4 for both)
3. Get intersection of the sets. (‘ht’)
4. Calculate:
   a. 2(len(intersection))/(len(A)+len(B))

Sørensen–Dice Coefficient of 0.25 or 25% similarity

SPEED

Sequence (length)	Levenshtein / Sorensen		Time (**Parallelized** | Non-parallelized)
E.Coli (4641652)	1	0.909091	**2.01** sec | 5.83 sec
Bacillus Anthracis (5227293)	1	0.909091	**2.19** sec | 6.46 sec
Bacillus Cereus (5411809)	1	0.909091	**2.27** sec | 6.60 sec

Substring: **ACTCACCACTGT**

CPU: Intel i7-4700HQ

MEMORY

K-mer generation

- Generate as many k-mers as possible
- Process data
  - Compare each k-mer to the substring in question
- Repeat until k-mers have been exhausted

Driver calculates the number of k-mers to generate based on sequence length and available memory, and scaled to sequence size

- Necessary to break up the volume of k-mers generated due to large sequence size

- Problem encountered due to differing sizes of sequences. Ex: If the max k-mers at a time is 50MB, but the E. Coli sequence is 2MB, we are requesting more k-mers than can be generated

GPU PARALLELIZATION

Idea:

- Generate X k-mers at a time
- Assign a **block of GPU threads**, 1 per k-mer
- Each thread calculates distance and assigns value to array

How:

- **OpenMP PTX offloading** technology

Complications:

- Need **special compilers**
- These compilers are too buggy

Result:

- Could process sequences on GPU but **could not assign the results**

GPU PARALLELIZATION

Attempt:

1. Set-up **build system** to support GPU offloading (not easy)
2. Explicitly **transfer data to GPU**
3. **Target** the GPU
4. Define **teams + thread blocks**
5. **Distribute threads** to C code

Example:

```
#pragma omp target data map(tofrom: ranks)
#pragma omp target teams
#pragma omp distribute parallel for
dist_schedule(static, 128)
for (k-mer in sequence) {
    ...
}
```

```
[ 18%] Linking CXX executable run
lto1: internal compiler error: bytecode stream: expected tag vec_unpack_float_hi_expr instead of LTO_UNKNOWN
0xab6e6b lto_tag_check
        ../../gcc/gcc/lto-streamer.h:1014
0xab6e6b streamer_read_tree_bitfields(lto_input_block*, data_in*, tree_node*)
        ../../gcc/gcc/tree-streamer-in.c:474
0x821ec1 lto_read_tree_1
        ../../gcc/gcc/lto-streamer-in.c:1330
0x822657 lto_read_tree
        ../../gcc/gcc/lto-streamer-in.c:1363
0x822657 lto_input_tree_1(lto_input_block*, data_in*, LTO_tags, unsigned int)
        ../../gcc/gcc/lto-streamer-in.c:1475
0x8228b9 lto_input_tree(lto_input_block*, data_in*)
        ../../gcc/gcc/lto-streamer-in.c:1492
0x79e783 inline_read_section
        ../../gcc/gcc/ipa-inline-analysis.c:4335
0x79e783 inline_read_summary()
        ../../gcc/gcc/ipa-inline-analysis.c:4394
0x883577 ipa_read_summaries_1
        ../../gcc/gcc/passes.c:2814
0x5a4380 read_cgraph_and_symbols
        ../../gcc/gcc/lto/lto.c:2913
0x5a4380 lto_main()
        ../../gcc/gcc/lto/lto.c:3308
Please submit a full bug report,
with preprocessed source if appropriate.
Please include the complete backtrace with any bug report.
See <https://gcc.gnu.org/bugs/> for instructions.
mkoffload: fatal error: /home/zach/offload/install/bin/x86_64-pc-linux-gnu-accel-nvptx-none-gcc returned 1 exit status
```

```
ach@zach-desktop:~/src/CS499-Bioinformatics/build$ nvprof ./run ../test/data/E-Coli.fna ACTGACGCAGACG
=12944== NVPROF is profiling process 12944, command: ./run ../test/data/E-Coli.fna ACTGACGCAGACG
=12944== Profiling application: ./run ../test/data/E-Coli.fna ACTGACGCAGACG
=12944== Profiling result:
            Type  Time(%)      Time     Calls       Avg       Min       Max  Name
GPU activities:   98.20%  186.31us         1  186.31us  186.31us  186.31us  _ZN10FuzzyMatch3RunERKNSt7__cxx1112basic_stringIcSt11char_traitsIcESaIcEEES7_b$_omp_fn$0
                   1.15%  2.1760us         2  1.0880us     992ns  1.1840us  [CUDA memcpy HtoD]
                   0.66%  1.2480us         1  1.2480us  1.2480us  1.2480us  [CUDA memcpy DtoH]
     API calls:   72.87%  261.31ms         1  261.31ms  261.31ms  261.31ms  cuCtxCreate
                  22.82%  81.841ms         1  81.841ms  81.841ms  81.841ms  cuCtxDestroy
                   1.47%  5.2751ms         1  5.2751ms  5.2751ms  5.2751ms  cuModuleLoadData
                   1.17%  4.1991ms        24  174.96us  33.223us  1.7427ms  cuLinkAddData
                   0.52%  1.8631ms         1  1.8631ms  1.8631ms  1.8631ms  cuLinkComplete
                   0.35%  1.2727ms         1  1.2727ms  1.2727ms  1.2727ms  cuMemAllocHost
                   0.32%  1.1627ms         3  387.58us  8.9020us  978.30us  cuMemFree
                   0.17%  592.48us         1  592.48us  592.48us  592.48us  cuMemFreeHost
                   0.10%  367.47us         3  122.49us  9.3770us  204.38us  cuMemAlloc
                   0.06%  232.20us         1  232.20us  232.20us  232.20us  cuLaunchKernel
                   0.05%  196.40us        14  14.028us     197ns  184.96us  cuDeviceGetAttribute
                   0.05%  188.97us         1  188.97us  188.97us  188.97us  cuCtxSynchronize
                   0.01%  36.384us         1  36.384us  36.384us  36.384us  cuLinkCreate
                   0.01%  22.786us         2  11.393us  7.5000us  15.286us  cuMemcpyHtoD
                   0.01%  19.676us         1  19.676us  19.676us  19.676us  cuMemcpyDtoH
                   0.00%  5.3990us         4  1.3490us     438ns  2.3980us  cuDeviceGetCount
                   0.00%  4.3350us        10     433ns     252ns     960ns  cuCtxGetDevice
                   0.00%  4.1650us         1  4.1650us  4.1650us  4.1650us  cuDeviceGetPCIBusId
                   0.00%  4.1510us         1  4.1510us  4.1510us  4.1510us  cuInit
                   0.00%  3.7720us         5     754ns     574ns  1.1650us  cuMemGetAddressRange
```

PROBLEMS

Team related

- **Budgeting time** among other classes for project
- **Differing knowledge** with the material and implementation
    - C++ familiarity
    - Domain knowledge

Project related

- **Platform support** for different environments
    - Windows, Mac OS, Linux distro.
- **Meaningful results** for combining Sorensen and Levenshtein

SUMMARY

- Overall, a **positive experience**
  - **Practiced** class material
  - **Collaborated** as a team
  - **Added value** because of optimizations and parallel computing
  - **Continuing** development

REFERENCES

Distance Metrics and FASTA format:
Wikipedia

GPU Parallelization:
https://www.openmp.org/wp-content/uploads/SC18-BoothTalks-Ozen.pdf