Solution to the Corteva Agriscience^{TM} Challenge[1]

Tianhao Wang, Sue Kim, Seamus Whoriskey

ABSTRACT

This paper describes in detail the methodology behind our solution to the Corteva Agriscience^{TM} Challenge.[1] The motivation behind this challenge is to improve upon the existing methods – by either increasing time efficiency and accuracy – for characterizing alleles in uncharacterized genomes, and comparing transcripts from those genomes to already categorized genomes to further our understanding about this less well-classified genome.

INTRODUCTION

This project was built with the Corteva Agriscience^{TM} Challenge[1] in mind. The goal of this challenge was to improve upon the current methods for finding the best mappings of different genomes and transcripts without sacrificing accuracy. To do this, our team utilized gffutils[2] to create a database for each genome provided using their respective gff3 files, which allowed us to understand the relationships between each type of sequence and generate database to provide fast access to each instance, we then fed these databases to minimap2[3] to align the transcripts from a source genome to chromosomes of a target genome, and perform further analysis base on the alignment information.

PIPELINE

Pre-processing

To manipulate the given gff3 files into a more manageable format, we used gffutils[2] to create a database from each gff3 file. gffutils essentially creates a MYSQL database, storing all the relationships between features in depth and we can easily access all the 'Parents' and 'Children' of each feature. Each feature can be obtained as an entry from their respective primary key. In our project, the primary key is set to be the ID and features, e.g., 'gene', 'mRNA','exon' and 'CDS' are all features accessible from their ID. The following graph explains the schema of the database. And we can also use the **region** function to retrieve features within the region.

```
CREATE TABLE features (
    id text,
    seqid text,
    source text,
    featuretype text,
    start int,
    end int,
    score text,
    strand text,
    frame text,
    attributes text,
    extra text,
    bin int,
    primary key (id)
    );
```

Figure 1. Database Schema

- inputs: chromosome.fasta, transcript.fasta,gff3 file for A,B and C chromosome
- output: results of mapping AB, BC, AC in .tsv format

SOFTWARE AND TOOLS

Minimap2

Minimap2[3] was chosen as our main alignment tool because of its ease of use, fast alignment speed and accurate results. By comparing the actual running results, we choose Oxford Nanopore read for our mapping.

For performing alignment, we set the source sequence as the transcript we are aligning, while the target sequence is set as the chromosome sequence. Due to the size of the chromosome file, we have to initialize a new aligner for each chromosome read. Each transcript sequence will be aligned to every chromosome and Minimap2 will return the 'best' reasonable hits. These hits are the most essential aspect in our implementation, by filtering out these hits we can obtain information to classify calls and categories.

Unfortunately, Minimap2 does not provide accuracy from its alignment, but we calculated a relevant accuracy from the information it provides. Minimap2 provides **blen, mlen** and **NM** as a measurement of the quality of the matching window. Both **blen** and **mlen** exclude the ambiguous matches, but the sequence reading contains plenty of read-errors and mapping a transcript to a different chromosome is noisy, so excluding ambiguous matches will result in missing too many accurate matches. Therefore, the accuracy is calculated by $accuracy = 1.0 - \frac{NM}{transcript\_length}$

- **blen**: length of the alignment, including both alignment matches and gaps but excluding ambiguous bases.,
- **mlen**: length of the matching bases in the alignment, excluding ambiguous base matches.
- **NM**: number of mismatches, gaps and ambiguous poistions in the alignment

Clustering

We used CD-HIT as our clustering tool to perform gene-level clustering to find the neighborhoods of genes, in another word, similar genes within same chromosome are clustered together as neighbors and will be used to detect gene fusions.

GffCompare

We used GffCompare[4] on each of the given gff3 files to merge duplicate transcripts within the same file, so that we did not analyze duplicate transcripts within the same genome multiple times, thus shortening the time taken to execute our solution.

METHODOLOGY

This section will dive further into detail about the methods used to detect and classify each call and category. The classification begins by filtering the hits based on if the accuracy of the respective hit exceeded one of two set thresholds. The hits above the higher accuracy threshold will be considered as candidates for **unique_transcript** and **multiple_transcript** while the other hits will then be classified as **absent_transcript**, **absent_gene** or **absent_genome**.

Unique_transcript

Candidates with accuracy above the first threshold will be considered as potential **unique_transcript** matches. We use the database generated in pre-processing to retrieve the target transcripts within the matching window. In the **mRNA-level**, if there is only one transcript retrieved and the window contains the entire domain of the target transcript, then we classify this transcript as **unique_transcript** and **exact_match**. Next, in the **exon-level** comparison, we classify the target transcript that contains all the internal exons of the source transcript in the matching window but with different marginal position as **unique_transcript** and **all_jxn_match**. **source_contained** and **target_contained** are also detected in a similar way.

Multiple_transcript

A transcript with multiple **unique_transcript** classifications will be output as **multiple_transcript**

Gene_fusion

gene_fusion can be detected when the matching region contains multiple transcripts with distinct parent genes. This method depends on the spatial locality of genes, in other words, if two genes are close to each other in space, then it is very likely that **gene_fusion** has occurred. We also tried to detect **gene_fusion** from the clustering neighbors by performing local alignment but the result was hard to interpret.

Absent_transcript

Candidates with accuracy below the first threshold but above second threshold will be considered as potential **absent_transcript** classifications. In the **mRNA-level**, if the match region contains at least one transcript then we continue to the **exon-level** to classify categories.

Absent_gene

Candidates that failed to be classified as **absent_gene** will be considered as potential **absent_gene**. In the **gene-level**, if the matching windows do not contain any gene, then it means the source transcript has similarity to the region, but the region is not an existing gene. Thus, **absent_gene** is detected

Absent_genome

Alignment hits below the second threshold will be classified as **absent_genome**

Ranking

The goal of this challenge is to find the best similarity between the source transcript and the target genome, where **unique_transcript** and **multiple_transcript** are the ideal cases. Therefore, we rank the calls to ensure our output only contains the best case among all the possible cases.

unique_transcript → **multiple_transcript** → **gene_fusion** → **absent_transcript** → **absent_gene** → **absent_genome**

CURRENT ISSUES AND FUTURE DEVELOPMENT

Due to the workload and time limitation, we could not able to make use of the neighbors we described above to work, but we believe the method behind is reasonable and capable of making a great improvement. There is still a lot of space for us to improve our score.

REFERENCES

[1] Corteva Agriscience™ Challenge: Methods for Determining Similar Sequences Across Genomes,
`https://www.innocentive.com/ar/challenge/9934185`
[2] gffutils documentation
`https://pythonhosted.org/gffutils/`
[3] minimap2 documentation
`https://github.com/lh3/minimap2/blob/master/README.md`
[4] GffCompare documentation
`https://ccb.jhu.edu/software/stringtie/gffcompare.shtml`