Agile Bioinformatics - Projects

1. UMIs collapsing

Description

Unique molecular identifiers (UMIs) are short random sequences attached to RNA molecules at the beginning of library preparation. They allow to identify duplicates of distinct RNA molecules and therefore remove PCR amplification bias. Reads with the same UMI and aligned to the same genomic regions can be collapsed to a single representative read. However, errors in base-calling of the UMI sequence causes increased number of unique sequences found in the sample and reduced precision of original transcripts abundance quantification.
The aim of the project is to prepare a CLI tool which will detect the UMIs sequences (without mapping to genome) and apply correction of the UMIs sequencing errors. The correction may be based on solutions provided in literature, any publicly available tool for UMIs detection or it may be novel approach (with documented justification). The method should be described in a documentation.
Optionally: add mapping reads (with any common tool), then include information about the mapping in the UMI detection and reads counting.

Input

Fastq file

Output

Processed fastq file, UMIs statistics

Literature and similar tools

- https://cgatoxford.wordpress.com/2015/08/14/unique-molecular-identifiers-the-problem-the-solution-and-the-proof/
- https://github.com/CGATOxford/UMI-tools/blob/master/doc/QUICK_START.md#step-3--extract-the-umis

Requirements

1. GitHub/Gitlab repository.
2. Project backlog.

3. Agile approach applied: sprints, reviews, planning.
4. Tests implemented.
5. Code review.
6. Documentation summarizing algorithms, methods and technologies.

2. BRowser Of VARiants

Description

VCF is a standard format of text file used for storing information about genomic variants. Browsing such files or extracting a particular piece of information requires specific tools. The aim of the project is to prepare a command line application which allows a user to:

- Filter the variants according to user defined rules (e.g. by location, type, length, content of a selected INFO field)
- Filter the variants by regions provided in a bed file
- Calculate basic statics on the selected variants
- Extract select fields to another format (e.g. TSV)

Optionally: Implement a simple GUI allowing a user to filter the VCF and browse the results.

Input

VCF file (e.g. any VCF from TCGA)
User query (e.g. as an YAML file)

Output

VCF statistics
Filtered VCF file

Literature and similar tools

- http://snpeff.sourceforge.net/SnpSift.html
- http://bioinformaticstools.mayo.edu/research/vcf-miner/
- VCF format: https://samtools.github.io/hts-specs/VCFv4.2.pdf

Requirements

1. GitHub/Gitlab repository.
2. Project backlog.
3. Agile approach applied: sprints, reviews, planning.

4. Tests implemented.
5. Code review.
6. Documentation summarizing algorithms, methods and technologies.

3. Genome SNP fingerprints

Description

Personal genomes analysis requires comparisons of big datasets of human genomes, characterize by features like SNPs sets. Such comparisons are needed for quality control reasons, identification of duplications or close relatives. Because of large number of genomic features which need to be accessed an algorithm which will simplify their description is needed.
Gulsman et al. proposed an ultrafast method of personal genomes' comparison. The method transforms a standard genome representation (e.g. list of SNPs) into a genome fingerprint. Reduction of the data size makes the comparison very fast but evaluation of the 'genomic distance' between samples is still possible. The aim of the project is to prepare a tool which calculates genomic fingerprints and genomic distances between provided samples. The tool should implement the approach published in the Gulsman's article.
Optionally:

1. *Extend the tool of creating a of samples, fingerprints and distances. The tool should allow a user to create and update the database.*
2. *Propose an extension of the tool with a method for calculating fingerprints on CNVs (instead of SNVs).*

Input

VCF files and sample metadata (e.g. from TCGA)

Output

Calculated fingerprints and a report of distances between provided samples.

Literature and similar tools

- https://www.ncbi.nlm.nih.gov/pmc/articles/PMC5623000/

Requirements

1. GitHub/Gitlab repository.
2. Project backlog.
3. Agile approach applied: sprints, reviews, planning.
4. Tests implemented.

5. Code review.
6. Documentation summarizing algorithms, methods and technologies.

4. Radiology Images Classification

Description

Images and their description are crucial in diagnostic pipelines. So far, most of the work was done to independently analyze text and pictures. But most predictive power should be in a combination of those. As a project, you will be requested to develop different algorithms based on Neural Networks to predict chosen labels from ROCO dataset. ROCO dataset contains radiological images, their description (caption) and tags. Some of the tags in this project will be treated as labels. As an outcome, you should provide an analysis of algorithms performance and code for them. Code for non-algorithm parts (like DataLoader etc.) should be covered in unit tests and algorithm training and testing procedure in regression tests. At least 4 algorithms should be compared, examples of them:

- Ensemble (e.g. using Random Forest) of results from CNN network (like ReSNeXt) for images and RNN network (like GRU) for text
- Concatenation of latent spaces of image and text representation based on autoencoders (one for an image, one for text)

As you should provide 4 algorithms, a part of the project is a literature review. So summarization of that literature should be included in the project outcome.
In business case code for algorithms and Data Loaders, etc., should be in .py files, analysis of results, obtaining results, training procedure can be either in .py files as Jupiter Notebooks. Algorithms and data preprocessing step are directly integrated to the application in which algorithm is used. Code should be reusable.
This project can be extended to publication if the results will be promising.
Tags to predict:

ct, tomography, radiograph, chest, lesion, tumor, xray, artery, mri, resonance, magnetic, bones, abdominal, abdoman, sagittal, lung, ultrasound

Literature

- https://github.com/razorx89/roco-dataset

Requirements

1. GitHub/Gitlab repository.
2. Project backlog.
3. Agile approach applied: sprints, reviews, planning.
4. Tests implemented.
5. Code review.
6. Documentation summarizing literature review, algorithms, methods and technologies.
7. Report from jupyter notebook in .pdf or .html summarizing the results.

5. Radiology Images Caption Generation (6 or 3 people)

Description

Description of the radiological image is widely used in diagnostic pipelines. Automation of this process should help to reduce time to obtain a diagnosis. Based on ROCO dataset, you should build a generative model, which is going to describe the images. .Then this model should be incorporated in the web application, which allows to load an image and obtain the caption of it. At least 3 different models should be compared before incorporating the algorithm. Algorithm example: https://arxiv.org/pdf/1711.07068.pdf
As you should provide 3 algorithms, a part of the project is a literature review. So summarization of that literature should be included in the project outcome.
In business case code for algorithms and Data Loaders, etc., should be in .py files, analysis of results, obtaining results, training procedure can be either in .py files as Jupiter Notebooks. Algorithms and data preprocessing step are directly integrated to the application in which algorithm is used. The code should be reusable.
If for 3 people, no app and 2 models.
This project can be extended to publication if the results will be promising.

Literature

- https://github.com/razorx89/roco-dataset
- https://arxiv.org/pdf/1711.07068.pdf

Input

Radiology image from ROCO dataset

Output

Caption for that image

Requirements

1. GitHub/Gitlab repository.
2. Project backlog.
3. Agile approach applied: sprints, reviews, planning.
4. Tests implemented.
5. Code review.
6. Documentation summarizing literature review, algorithms, methods and technologies.
7. Report from jupyter notebook in .pdf or .html summarizing the results.

6. Cancer Tissue Segmentation

Description

Localizing the cancer tissue on the histopathology image is a key for proper diagnosis. You goal here is to develop 2 algorithms:

- Segmentation algorithm based on lesions annotations
- Segmentation algorithm based only on label (present/not)
    - So in training you are not using the annotations, but only labels that tissue is presence. Since images are huge, they need to be splitted into patches (sliding window mechanism). Then based on segmentation for training you are able to generate patches containing some portion of cancer tissue.

Sliding window mechanism is a common thing for both algorithms. As a part of the project you need to check current State of the Art for weakly labeled supervised segmentation and choose one of the algorithms to implement. The second algorithm should be re-implemented with your ideas based on Chameleon winners (see: Literature section) or replace with different one (e.g. U-Net, we recommend U-Net due to the short training time).
In business case code for algorithms and Data Loaders, etc., should be in .py files, analysis of results, obtaining results, training procedure can be either in .py files as Jupiter Notebooks. Algorithms and data preprocessing step are directly integrated to the application in which algorithm is used. The code should be reusable.

As a results also report from the results should be generated, which will compare how good algorithm trained without annotations is to algorithm with annotations.
This project can be extended to publication if the results will be promising.

Literature

- https://camelyon17.grand-challenge.org/Data/
- https://arxiv.org/pdf/1703.02442.pdf
- https://arxiv.org/pdf/1505.04597.pdf
- https://arxiv.org/pdf/1806.04659.pdf

Requirements

1. GitHub/Gitlab repository.
2. Project backlog.
3. Agile approach applied: sprints, reviews, planning.
4. Tests implemented.
5. Code review.
6. Documentation summarizing literature review, algorithms, methods and technologies.
7. Report from jupyter notebook in .pdf or .html summarizing the results.