DrivenData Genetic Attribution Challenge 2020

DEMOCRATIZING GENETIC ATTRIBUTION

DrivenData Username: VictorCallejas

Submission Timestamp: 2020-09-30 18:06:58 UTC
Public Top 10 Accuracy: 0.87
Private Top 10 Accuracy: 0.84

Email: vcallejasfuentes@gmail.com

November 2, 2020

Abstract

This paper proposes a new technique for the genetic attribution problem. Tools as BLAST, CNNs, Transformers... have been used for this problem with good results but all of them are computationally heavy and need of a lot of computing power and time to train the models, which makes this field inaccesible for most people.

In this paper I propose a new technique based on the field of Natural Language Processing to tackle this problem. This method has surpassed the BLAST benchmark by 9 points of Top-10 accuracy, it can run on a laptop within a few minutes, which makes it very accessible.

1 Problem statement

Current models used in this field as BLAST, are **computationally heavy and not accesible for most people**.

For other models I have developed for this competition, I have generated features with BLAST. With an instance which has 96 vCPU and 192 GiB RAM in Amazon Web Services, it took almost **two days to run BLAST tool** over a 5 fold with competition data with **a cost of 423,43 dollars**.

This is by no means accesible for a lot of researchers and has some great disadvantages for the field. For example, if I wasn't an university student which has free credits on AWS, it would have been impossible for me to train a model with BLAST features, and of course I would not feel encouraged research on this field.

Other fields inside Machine Learning have experimented in the past huge development once they have been made more accesible.

Also this models are **difficult to retrain** as they need lots of time and computing power. This makes **finetunning very difficult** . For example, if I would like to finetune this model for my specific task, imagine I do not want to classify between 1314 laboratories, I just need to classify between 40 laboratories. With previous models I would need to recreate BLAST fasta database or finetune a transformer model. This could constrain the access for some business in the industry and increase costs. Also in the case exposed above, **finetunning the model could increase drastically the accuracy**.

Other models such as random forest over generated N-GRAM features have been proposed. The benchmark for it in this challenge is 33 points of Top 10 accuracy. If more N-GRAMS are generated the score improves to a limit. The problem is that the **N-GRAM features are generated by permutations of the DNA bases (A,T,G,C,N). The cost of permutations increases exponentially** with the number of bases(b), length(l) and linearly with the number of different lenghts you want to generate(n).

$$P = \frac{b!}{b - l!} \cdot n \tag{1}$$

Most of this features lack of meaning and therefore do not improve the model.

2 Proposed Architecture

The proposed architecture is very **simple and lightweight**, it consists on a **one layer dense neural net over features extracted with a tokenizer**.

2.1 Tokenizing DNA sequences

Tokenizers[1] are a tool of the Natural Language Processing field used to preprocess sentences or documents into a sequence of ids(unique numbers) which later on a neural network, usually transformers[2], processes.

Tokenizers are trained over a corpus of text, and from it you obtain a vocabulary, that maps the tokens(words, subwords...) to ids.

In our case **we train the tokenizer over all DNA sequences**, and generate our vocabulary of dna subsequences, which is far richer than N-GRAM permutations.

2.2 Model

Usually the tokenized sentence is passed to a transformer model which is state of the art in NLP. These models preserve positional information of the tokens in the sequence, so they are able to learn more complex relationships. But these architectures requiere high computing power, very long training times and lots of data. These transformers architectures have a mechanism called attention which scales cuadratically with the sequence lenght, so **it is very difficult to use these models with dna sequences**. Other models have been proposed to tackle this problem as (Linformer[3], Reformer[4]...), but **still the spatial complexity is very high**.

We propose to use directly the ids extracted by the tokenizer by making a **one hot encoding representation** of the count of the ids.

This representation has the lenght of the tokenizer vocabulary size and is concatenated with the tabular data of the sequences.

Over these extracted features multiple models can be trained as SVM, Decission Trees, Linear Models, Neural Networks... Depending on the model, **scaling the features could be recommended**,

We choose to train a **neural network of just one dense linear layer**. **It takes the tabular data and sequence tokenized one hot representation and outputs the probability for each laboratory**.

With this model we achieved the following results.

	Top 10 Accuracy	Accuracy
Private	0.84	-
Public	0.87	-
Validation	0.92	0.78
Train	0.99	0.95

This model can be trained in a laptop in a matter of minutes.

These rich tokenizer features allows fast iteration, easy interpretability, finetunning and cost effective training allowing more people, business and institutions to join the field of genetics.

References

[1] SentencePiece: A simple and language independent subword tokenizer and detokenizer for Neural Text Processing https://arxiv.org/abs/1808.06226

[2] Attention Is All You Need https://arxiv.org/abs/1706.03762

[3] Linformer: Self-Attention with Linear Complexity https://arxiv.org/abs/2006.04768

[4] Reformer: The Efficient Transformer https://arxiv.org/abs/2001.04451