## **Abstract**

**Motivation:** Infectious diseases from novel viruses are becoming a major public health concern. Increased interaction between human and nature, increased contact between human and wild animals increase the risk of apparition of new virus and the risk of interspecies transmission.

In this work, we have focused on two key aspects related to RNA and proteins evolution and interaction.

Virus mutation is the first aspect by identifying virus sequences mutation, tracking these sequence mutations then forecasting potential mutation area in the RNA sequence.

We have combined the RNA alignment sequence method with a clustering algorithm and neural network in order to develop a novel method to insure the tracking and detect new location of the mutation.

Virus-Host interactions is the second key aspect, to forecast potential proteins and genes which can interact with a Virus and therefore being able to detect potential gene impacted and potential species impacted by the Virus.

Current computational prediction methods for novel viruses are based only on protein sequences. We have implemented a method allowing to combine protein sequence analysis together with Disease phenotypes (i.e., symptoms).

**Results:** We developed a tool and an application which has two major functionalities. The tool can predict and recognize the origin of a Virus RNA sequence and the tool can allow it to detect new mutation locations in the sequence. Also the tool is able to perform protein alignment and homogeneity calculation, a method which is associated with a Phenotype analysis inspired by Natural language recognition techniques.

## **1 Introduction**

Infectious diseases emerging unexpectedly from novel pathogens have been a major public health concern around the globe. Pathogens disrupt host cell functions and target immune pathways through complex inter-species interactions of proteins and DNA. As well, evolution of the pathogen in form more or less lethal for infected species is a key aspect which impacts the research of diagnosis, treatment and vaccine. The study of pathogen mutation and pathogen–host interactions (PHI) can therefore support the research of novel therapeutics or support the search for reuse of already available drugs.

We developed a tool and associated applications allowing 2 key functionalities:

1. Identification of mutation, localization of the mutation and tracking of the mutation on a virus, based on RNA sequencing. A reference RNA sequence is used, then a database of RNA sequences is compared to the reference RNA sequence using the **Smith–Wa­ter­man algorithm** algorithm. From these alignments, we extract the mutation/mutation quantitty/mutation type and mutation location these information are used to to create a knowledge base. This knowledge base is used to teach a neuromorphic neural network.

Then a new sequence can be injected to the neural network, which will determine the Geography/Generation and Species of the tested sequence.

We tested this method on the Co-Vid19, but this method is genetic and can be used for any type of RNA sequences.

1. Identification of potential hosts of a virus by predicting the potential genes which are affected by a virus / pathogen, as well detection which are the most probable protein-protein interaction. We here combines two independent techniques:
   1. Alignment and comparison of proteins sequences of proteins from the virus / pathogen with a large database containing sequences of proteins from multiple species. Here the Needleman-Wunsch algorithm is mainly used to evaluate the homogeneity of the sequences
   2. We apply Natural Language recognition techniques to Phenotypes, so we could evaluate the relationship between virus phenotypes and phenotypes related to species and genes. Implementation of techniques such as Wor2Vect on existing phenotypes database allows to create important corpus, which can be used to define

Our method was inspired by the research ‘Prediction of novel virus–host interactions by integrating clinical symptoms and protein sequences’ from Wang Liu-Wei1 , Senay Kafkas , Jun Chen , Jesper Tegnér and Robert Hoehndor. Where we implemented different types of algorithms to perform the correlation task.

## **2 Materials and methods**

### **2.1 Data sources, phenotypes, ontologies**

2.1.1 Protein interactions prediction function

We used some of the major existing databases for phenotypes and proteins sequence as well as background information.

1. Interactions between hosts and pathogens were downloaded from the Host Pathogen Interaction Database (HPIDB), which contains several thousands of distinct pairs of protein-protein interaction between human and viruses, equipped with a corresponding MIscore
2. The phenotypes associated with pathogens were collected from the PathoPhenoDB.
3. The phenotypes associated with human genes were collected from the Human Phenotype Ontology (HPO)
4. Phenotypes associated with mouse genes and the orthologous gene mappings from mouse genes to human genes, originated from the Mouse Genome Informatics (MGI) database
5. Background knowledge from biomedical ontologies of phenotypes and GO classes:

Virus and family taxon from the NCBI taxonomy went obtained in OWL format and cross species went obtained from PhenomeNET Ontology in OWL format..

2.1.2 Virus RNA tracking

We use sequences provided by the NCBI database.

The application has been developed with Python and Neuromorphic neural networks are handled by a dedicated tool which is using Xilinx FPGA. Background knowledge has been created using groovy.

### **2.2 Application RNA Virus sequence tracking**

As a first step, we collect sequences from the NCBI database.

We made a use case on Covid 19 sequences. Each sequence constraints about 3000 nucleotides. As a first step, we perform the Local **Smith–Wa­ter­man algorithm** algorithm to perform sequence alignment.

**Description of the Smith–Wa­ter­man algorithm algorithm:**

The **Smith–Wa­ter­man algorithm** per­forms local [se­quence align­ment](https://wiki2.org/en/Sequence_alignment); that is, for de­ter­min­ing sim­i­lar re­gions be­tween two strings of [nu­cleic acid se­quences](https://wiki2.org/en/Nucleic_acid_sequence) or [pro­tein se­quences](https://wiki2.org/en/Protein_sequence). In­stead of look­ing at the [en­tire](https://wiki2.org/en/Needleman–Wunsch_algorithm) se­quence, the Smith–Wa­ter­man al­go­rithm com­pares seg­ments of all pos­si­ble lengths and [op­ti­mizes](https://wiki2.org/en/Mathematical_optimization) the [sim­i­lar­ity mea­sure](https://wiki2.org/en/Similarity_measure).

The al­go­rithm was first pro­posed by [Tem­ple F. Smith](https://wiki2.org/en/Temple_F._Smith) and [Michael S. Wa­ter­man](https://wiki2.org/en/Michael_S._Waterman) in 1981. Like the [Needle­man–Wun­sch al­go­rithm](https://wiki2.org/en/Needleman–Wunsch_algorithm), of which it is a vari­a­tion, Smith–Wa­ter­man is a [dy­namic pro­gram­ming](https://wiki2.org/en/Dynamic_programming) al­go­rithm. As such, it has the de­sir­able prop­erty that it is guar­an­teed to find the op­ti­mal local align­ment with re­spect to the scor­ing sys­tem being used (which in­cludes the [sub­sti­tu­tion ma­trix](https://wiki2.org/en/Substitution_matrix) and the [gap-scor­ing](https://wiki2.org/en/Gap_penalty) scheme). The main dif­fer­ence to the [Needle­man–Wun­sch al­go­rithm](https://wiki2.org/en/Needleman–Wunsch_algorithm) is that neg­a­tive scor­ing ma­trix cells are set to zero, which ren­ders the (thus pos­i­tively scor­ing) local align­ments vis­i­ble. Trace­back pro­ce­dure starts at the high­est scor­ing ma­trix cell and pro­ceeds until a cell with score zero is en­coun­tered, yield­ing the high­est scor­ing local align­ment. Be­cause of its qua­dratic com­plex­ity in time and space, it often can­not be prac­ti­cally ap­plied to large-scale prob­lems and is re­placed in favor of less gen­eral but com­pu­ta­tion­ally more ef­fi­cient al­ter­na­tives such as (Gotoh, 1982), (Altschul and Er­ick­son, 1986), and (Myers and Miller, 1988).

## Motivation of the **Smith–Wa­ter­man algorithm:**

In re­cent years, [genome pro­jects](https://wiki2.org/en/Genome_project) con­ducted on a va­ri­ety of or­gan­isms gen­er­ated mas­sive amounts of se­quence data for genes and pro­teins, which re­quires com­pu­ta­tional analy­sis. Se­quence align­ment shows the re­la­tions be­tween genes or be­tween pro­teins, lead­ing to a bet­ter un­der­stand­ing of their ho­mol­ogy and func­tion­al­ity. Se­quence align­ment can also re­veal [con­served do­mains](https://wiki2.org/en/Conserved_sequence) and [mo­tifs](https://wiki2.org/en/Sequence_motif).

One mo­ti­va­tion for local align­ment is the dif­fi­culty of ob­tain­ing cor­rect align­ments in re­gions of low sim­i­lar­ity be­tween dis­tantly re­lated bi­o­log­i­cal se­quences, be­cause mu­ta­tions have added too much 'noise' over evo­lu­tion­ary time to allow for a mean­ing­ful com­par­i­son of those re­gions. Local align­ment avoids such re­gions al­to­gether and fo­cuses on those with a pos­i­tive score, i.e. those with an evo­lu­tion­ar­ily con­served sig­nal of sim­i­lar­ity. A pre­req­ui­site for local align­ment is a neg­a­tive ex­pec­ta­tion score. The ex­pec­ta­tion score is de­fined as the av­er­age score that the scor­ing sys­tem ([sub­sti­tu­tion ma­trix](https://wiki2.org/en/Substitution_matrix) and [gap penal­ties](https://wiki2.org/en/Gap_penalty)) would yield for a ran­dom se­quence.

An­other mo­ti­va­tion for using local align­ments is that there is a re­li­able sta­tis­ti­cal model (de­vel­oped by Kar­lin and Altschul) for op­ti­mal local align­ments. The align­ment of un­re­lated se­quences tends to pro­duce op­ti­mal local align­ment scores which fol­low an ex­treme value dis­tri­b­u­tion. This prop­erty al­lows pro­grams to pro­duce an [ex­pec­ta­tion value](https://wiki2.org/en/Expectation_value) for the op­ti­mal local align­ment of two se­quences, which is a mea­sure of how often two un­re­lated se­quences would pro­duce an op­ti­mal local align­ment whose score is greater than or equal to the ob­served score. Very low ex­pec­ta­tion val­ues in­di­cate that the two se­quences in ques­tion might be [ho­mol­o­gous](https://wiki2.org/en/Homology_(biology)), mean­ing they might share a com­mon an­ces­tor.

## 

## 

## 

## 

## 

## **Explanation of the Smith–Wa­ter­man al­go­rithm**

Smith–Wa­ter­man al­go­rithm aligns two se­quences by matches/mis­matches (also known as sub­sti­tu­tions), in­ser­tions, and dele­tions. Both insertions and dele­tions are the operations that in­tro­duce gaps, which are rep­re­sented by dashes. The Smith–Wa­ter­man al­go­rithm has sev­eral steps:

1. **Determine the substitution matrix and the gap penalty scheme**. A substitution matrix assigns each pair of bases or amino acids a score for match or mismatch. Usually matches get positive scores, whereas mismatches get relatively lower scores. A gap penalty function determines the score cost for opening or extending gaps. It is suggested that users choose the appropriate scoring system based on the goals. In addition, it is also a good practice to try different combinations of substitution matrices and gap penalties.
2. **Initialize the scoring matrix**. The dimensions of the scoring matrix are 1+length of each sequence respectively. All the elements of the first row and the first column are set to 0. The extra first row and first column make it possible to align one sequence to another at any position, and setting them to 0 makes the terminal gap free from penalty.
3. **Scoring**. Score each element from left to right, top to bottom in the matrix, considering the outcomes of substitutions (diagonal scores) or adding gaps (horizontal and vertical scores). If none of the scores are positive, this element gets a 0. Otherwise the highest score is used and the source of that score is recorded.
4. **Traceback**. Starting at the element with the highest score, traceback based on the source of each score recursively, until 0 is encountered. The segments that have the highest similarity score based on the given scoring system is generated in this process. To obtain the second best local alignment, apply the traceback process starting at the second highest score outside the trace of the best alignment.

**Example of alignments:**

G T T - A C

| | | | |

G T T G A C

From the alignment we can identify the mutation location, type and quantity. This alignment is done in case of COVID 19 on sequences of about 30000 nucleotides.

**Classification of Mutation:**

Mutation are usually classified under the following categories:

**Missense mutation**

This type of mutation is a [change in one DNA base pair](https://ghr.nlm.nih.gov/primer/illustrations/missense.jpg) that results in the substitution of one amino acid for another in the protein made by a gene.

**Nonsense mutation**

A [nonsense mutation](https://ghr.nlm.nih.gov/primer/illustrations/nonsense.jpg) is also a change in one DNA base pair. Instead of substituting one amino acid for another, however, the altered DNA sequence prematurely signals the cell to stop building a protein. This type of mutation results in a shortened protein that may function improperly or not at all.

**Insertion**

An [insertion](https://ghr.nlm.nih.gov/primer/illustrations/insertion.jpg) changes the number of DNA bases in a gene by adding a piece of DNA. As a result, the protein made by the gene may not function properly.

**Deletion**

A [deletion](https://ghr.nlm.nih.gov/primer/illustrations/deletion.jpg) changes the number of DNA bases by removing a piece of DNA. Small deletions may remove one or a few base pairs within a gene, while larger deletions can remove an entire gene or several neighboring genes. The deleted DNA may alter the function of the resulting protein(s).

**Duplication**

A [duplication](https://ghr.nlm.nih.gov/primer/illustrations/duplication.jpg) consists of a piece of DNA that is abnormally copied one or more times. This type of mutation may alter the function of the resulting protein.

**Frameshift mutation**

This type of mutation occurs when the addition or loss of DNA bases changes a gene's reading frame. A reading frame consists of groups of 3 bases that each code for one amino acid. A [frameshift mutation](https://ghr.nlm.nih.gov/primer/illustrations/frameshift.jpg) shifts the grouping of these bases and changes the code for amino acids. The resulting protein is usually nonfunctional. Insertions, deletions, and duplications can all be frameshift mutations.

**Repeat expansion**

Nucleotide repeats are short DNA sequences that are repeated a number of times in a row. For example, a trinucleotide repeat is made up of 3-base-pair sequences, and a tetranucleotide repeat is made up of 4-base-pair sequences. A [repeat expansion](https://ghr.nlm.nih.gov/primer/illustrations/repeatexpansion.jpg) is a mutation that increases the number of times that the short DNA sequence is repeated. This type of mutation can cause the resulting protein to function improperly.

**Neural networks applied to the aligned sequence:**

The tool is then identifying the following to create the input vectors of the neural networks :

1. Location of the mutation
2. Quantity of mutation at each location
3. Type of the mutation.

A clustering algorithm is then used to generate a vector which can be used for a neural network.

The sequence of about 30000 nucleotides is then splitted in slice of 500 nucleotides each, 3 contexts are created:

i) First context with the center of location cluster mutation for each slices,

ii) Second context with the quantity of mutation for each slices

iii) Third context with the type of mutation for each slices

Created vectors are then used to train a neuromorphic neural network embedded in a FPGA Xilinx and by similar process, new vectors coming from new sequences can be tested.

Each of the vectors are characterized by a category number which represents the information on Geographical location, Species and period of recording of the sequence.

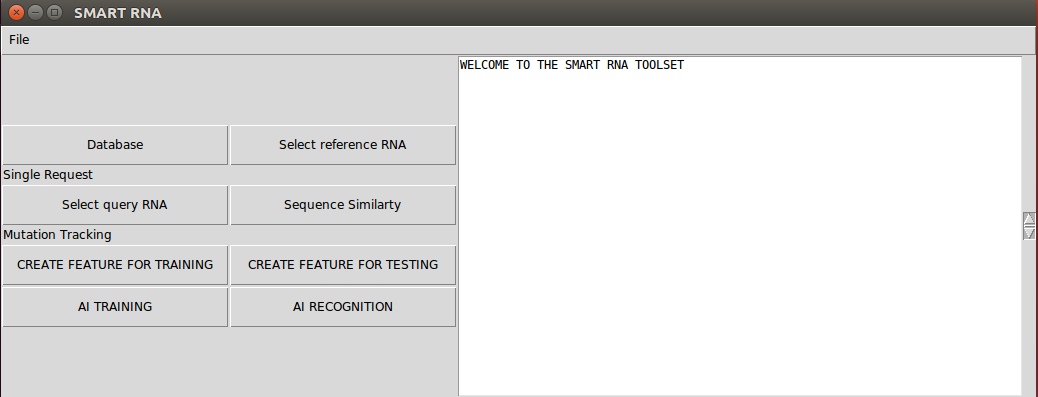
When testing a new vector to the neural network, the tool reports the category recognized allowing it to associate to the Geographical location, Species and period of recording of the sequence.

**Description of the tool and example**

The application has two main windows, the first one which is described below is used for RNA sequence tracking.

When opening the tool, the following windows does appear.

On the left side, we have the control panel and on the right side a text panel showing information and results,



The first button, ‘Database’ will open directly the link of the NCBI database, allowing to download RNA sequences of viruses

After downloading the sequences, it is needed to change their name with the following format:

1. Geographical localization ‘EU’ for Europe, ‘US’ for North America, INDIA, ASIA, CHINA…
2. The species ie HUMAN
3. The period of the sequence ie 1 or 2 or 3… corresponds to a period when the sequence was acquired.

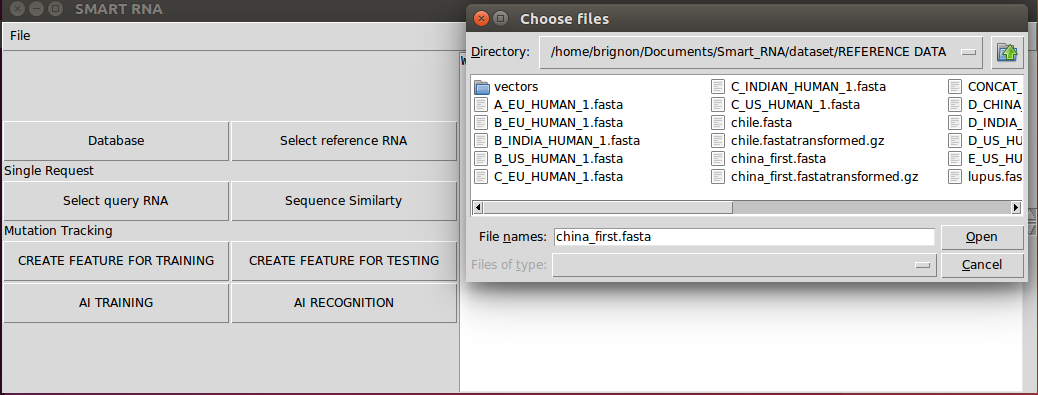
The naming is very important as it will be used to automatically annotate the training set, so no other actions are needed from the operator.

The triplet Geographical location + Species + Period is then converted to a category number automatically by the tool. These categories and conversion tables can be easily updated in the Python script without previous knowledge of Python programming.

**Reference RNA sequence:**

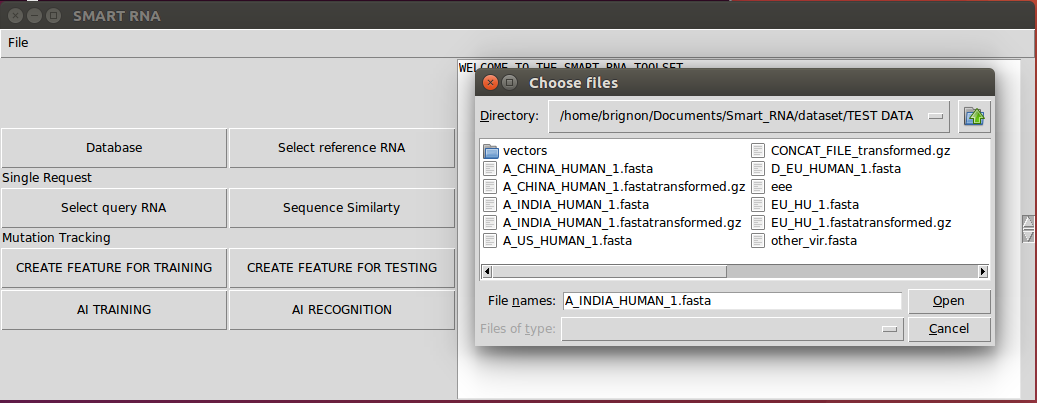
Pior any other actions, it is important to select the reference RNA sequence, against which other sequences will be compared.

This selection is done by using the button ‘Select reference RNA’.

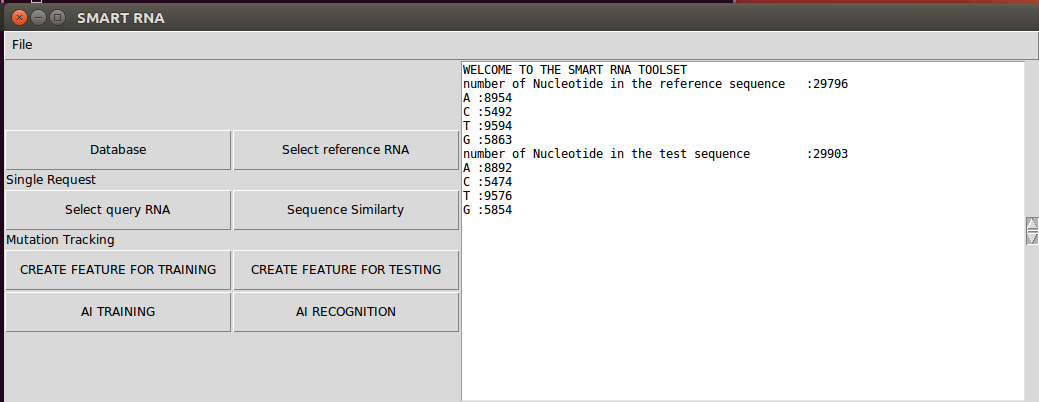
.

**Example of simple request:**

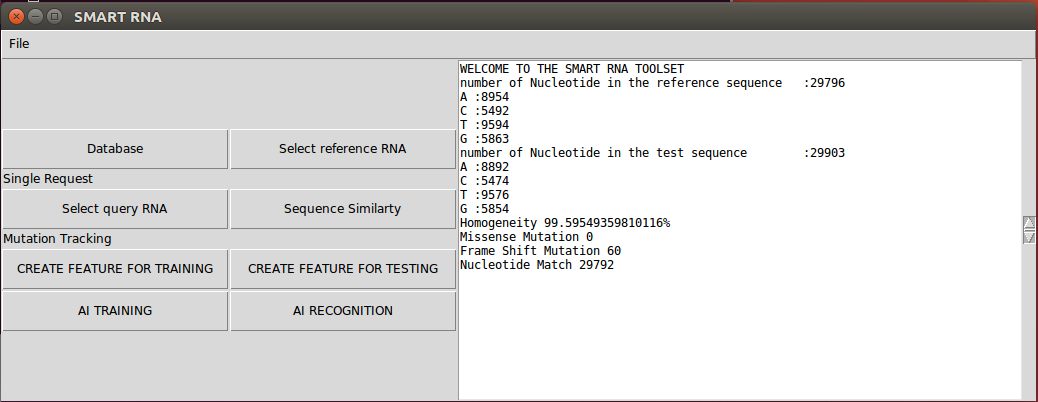
In case of simple request, we will select the query sequence by using the button ‘Select query RNA”



As the result of this selection, the tool is first automatically calculating the number of nucleotides of each sequence and the repartition between nucleotide types.



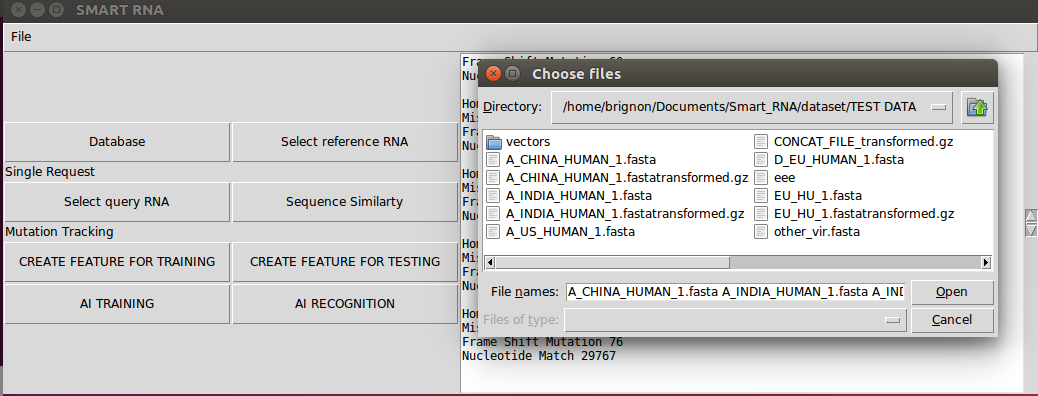
After the selection of the reference sequence and the selection of the query sequence, by using the button ‘Sequence Similarity’, the homogeneity between the two sequences will be calculated following the alignment of the sequences. The tool determines the definition of the quantity and type of mutation between the two sequences.



We can then see the result of the calculation appearing in the text area, with in this case and comparison between a sequence from India used as a query sequence and the first sequence reported from China of Covid a high homogeneity above 99,5% and 60 frameshift mutation.

**Sequence Mutation tracking with neural networks:**

To perform the tracking of the mutation, we first need to select the reference RNA sequence as described in above and then we need to constitute a training set. For doing this, we use the button ‘Create Feature for Training’, We can then select multiple sequences, which will be used to generate the set of vectors. These vectors will be part of the training set..

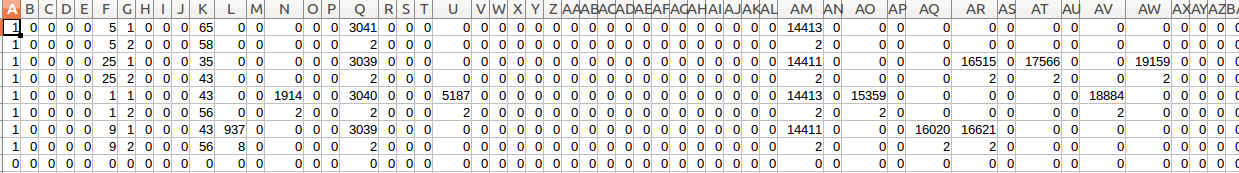


The following process will takes place for each sequence:

1. Alignment against the reference sequence
2. Calculation of the homogeneity against the reference sequence
3. Identification of the mutation (type, location, number)
4. Clustering of the mutation and information generated in vectors of 3 different contexts
5. Creation of a matrix containing the vectors of the training set.

The Matrix of features for the training set (see below) is containing key information such as the category of the sequence, the context number and the features

This matrix is afterward directly used to automatically train the neural network embedded in the FPGA.



To create a test set, similar steps and process are used, by using the button ‘Create Feature for Testing’

In this case, a matrix of feature is also documented, where category is not filled as providing the category will be the output of the neural network

**Training of the network**

The training of the network is done by using the button ‘AI Training’. It is then requested to select the matrix file which has been generated and composed of the feature set (see above)

The tool will automatically train the neuromorphic neural network, which is mainly implemented as a RCE-RBF algorithm (Restricted Coulomb Energy / Radial Basis Function)

**Test of unknown sequence and result presentation:**

By using the button ‘AI recognition’ we can select the test matrix which will be injected to the neural network.

The tool is then reporting the result of the recognition by providing:

* The category detected, automatically translates according to the conversion tables to geography, species and period.
* The context detected against the 3 contexts we have in the database.
* The distance to the center of the recognized neuron, which can provide a level of certainty.

**Example:**

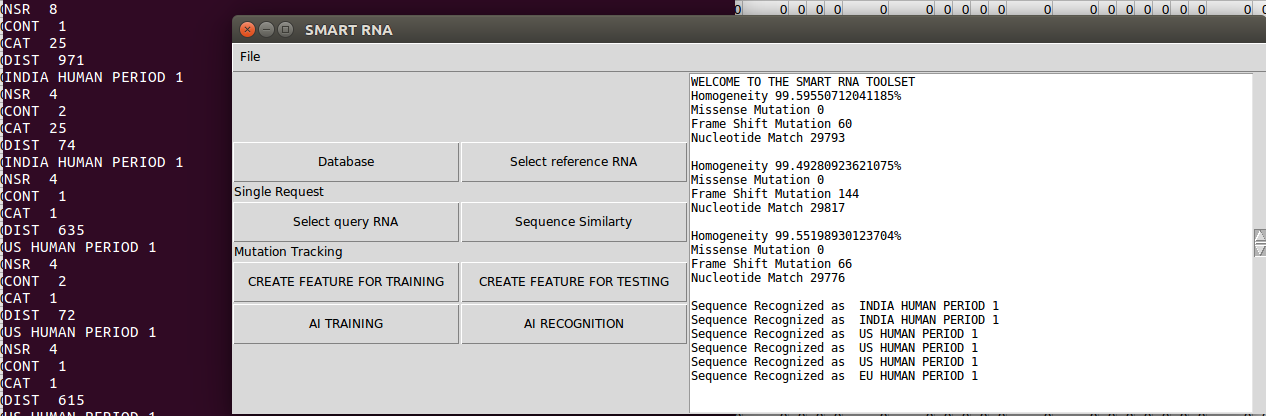
We realized an example using a reference sequence, the first COVID 19 RNA sequence coming from China and available on the NCBI database. We limited the training set to 6 sequences composed as follow:

* 2 sequences from India
* 2 sequences from Europe
* 2 sequences from US

We then injected as testing set 3 sequences composed as follow:

* 1 sequence from india
* 1 sequence from US
* 1 sequence from EU

Below are the result generated by the tool:



Explanation of the result:

NSR: NSR express if one category only triggered, if several categories triggered or if none of the categories triggered.

A value NSR=8 means that only one category triggered.

A value NSR=4 means that at least two categories are triggered.

A value NSR=0 means that none of the categories triggered.

CONT: CONT expresses the context, in our case we have 3 active contexts, but only 2 active in the test. So for each sequence from the testing set, a search is done in the context 1 and then in the context 2.

CAT: CAT expresses the category, it is represented by a number which is then translated into text based on a pre-established correspondence table.

DIS: DIST expresses the distance to the center of the category which triggered. In case several categories are triggered, the one with the smallest distance is reported by the tool.

The text box of the application is providing the result mentioning which category has been detected after the translation using the correspondence table. For each sequence, we have then 2 results corresponding to the 2 contexts.

Recognition result for the sequence India: Result is the expected one

The tool recognized for the context 1 as a sequence from India : Result is the expected one

The tool recognized for the context 2 as a sequence from India : Result is the expected one

Recognition result for the sequence US: Result is the expected one

The tool recognized for the context 1 as a sequence from US : Result is the expected one

The tool recognized for the context 2 as a sequence from US : Result is the expected one

Recognition result for the sequence EU: Result is uncertain

The tool recognized for the context 1 as a sequence from US : Result is not the expected one

The tool recognized for the context 2 as a sequence from EU : Result is the expected one

The conclusion in regards to sequences from India and the US would be clear, the tool has defined the proper origin of the virus.

The conclusion in regards to sequences from EU would be unclear. The tool has defined two potential origins of the virus and the result of the two contexts are not aligned.

**Summary of RNA sequence tracking**

In the example above, we show the capacity of the tool to track the localization / period / species of the virus. Extended demonstration could be done with support of researchers using a more important training set.

The tool returns 5 times out of 6 (about 83%) the expected result and did not provide the expected result in one case.

To further investigate this case, 3 direction could be taken:

* Enrich the training set (which was limited in this example) by adding as much sample as possible.
* Establish more complex arbitration logic by adding for example also in this logic the top 5 of the categories triggering.
* Considering a potential new mutation of the virus which went not yet taken into account and which could lead to such a result. This could be confirmed if several samples would also report the same result.

The tool is also able to report novelty. Novelty will be reported when no category triggered. Thisl novelty is important as it can bring information of unknown sequences with mutation in different locations and different quantities. The analysis of this novelty can allow us to complete the database and its annotation and to re-teach the neural network with additional information.

As additional use of the tool, the matrix created for the training set / testing set allows to draw charts showing the evolution of the position of the mutation in the sequences as well as the evolution of the quantity of the mutation in the sequence.

This information can forecast the risk of oncoming mutation and identify areas of the RNA which are more sensitive to mutation.

As an application of the tool as part of a concrete system, we could propose to connect this tool to next-generation sequencing systems (NGS). NGS are tool allowing to extract RNA sequence from sample, therefore combined with our tool, it could offer additional information in order to track origin/mutation of virus and report individual information of each sample to a complete central database allowing to better control epidemies (protection measure and treatment) by better knowing which branch of the virus is affecting precisely each location and each person.

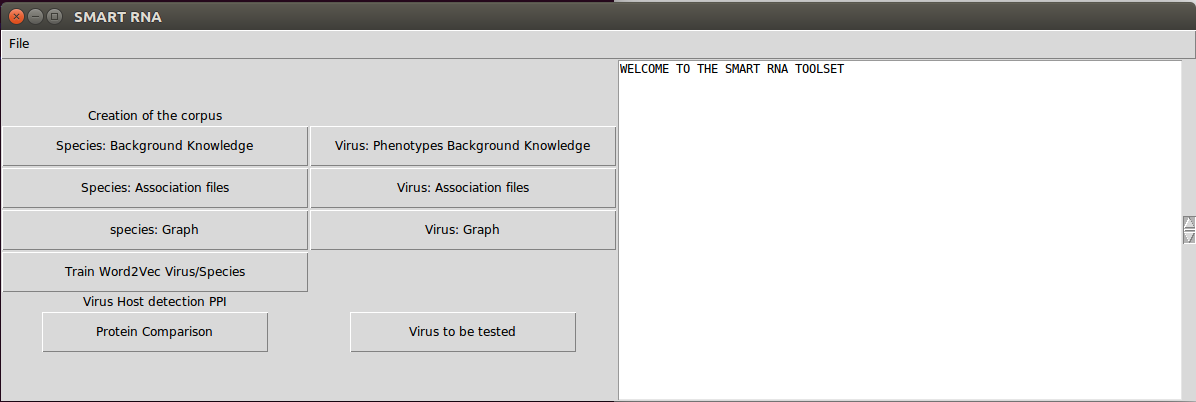
Virus are muting quickly and can change to form less or more severe which could also need different type of therapeutic. Therefore, this combination of NGS with our tool would make great sense to provide complete information about the disease on a certain patient and centralize additional information from the disease.

### **2.3 Protein-Protein Interaction / Virus host detection**

The second function of the tool is to establish correlation between a virus and the potential host gene and host proteins.

We apply two independent methods to perform this correlation.

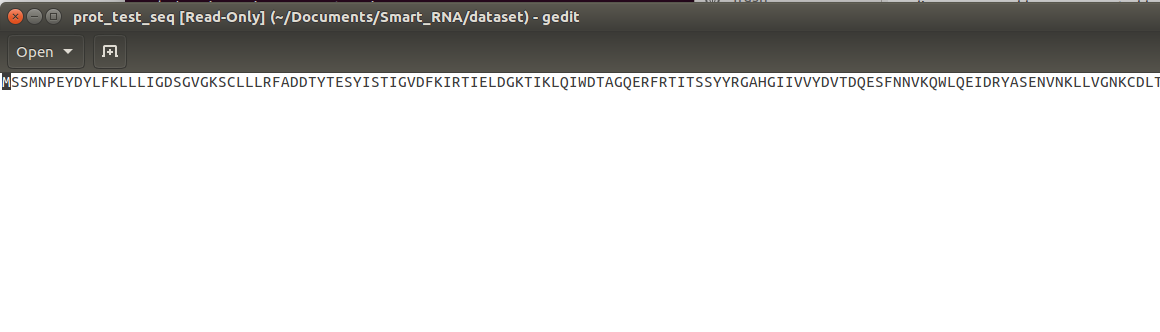
To perform this second function we utilize the below button and acquire the result in below text window.



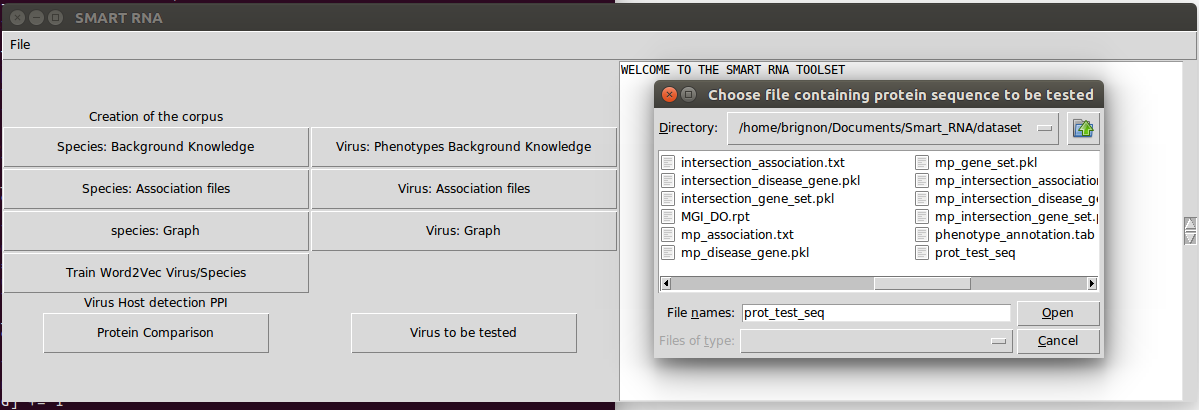
**Protein sequence comparison:**

The first method is the protein/protein sequence comparison. We apply the **Smith–Wa­ter­man algorithm** at protein sequence level to search for protein homogeneity between the protein sequence of the pathogens and a complete database of human/species protein sequence.

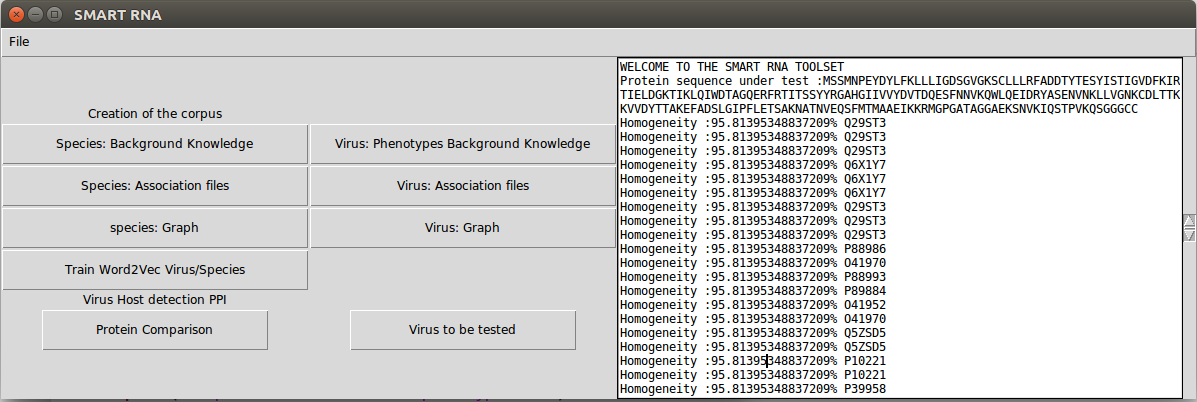
We first input the protein sequence in a text file and store it. This file will be used as an input by the tool. This allows flexibility to test any virus with known protein sequence.



In the tool, we then load the file with the protein sequence of the pathogen.



The tool will then perform sequence analysis and homogeneity analysis, aligning and comparing the input sequence with the complete database HPIDB.



The tool is then output reporting the Proteins which shows the highest homogeneity. The threshold can be configured, in this example it was set up a minimum homogeneity of 95.8%

**Phenotype analysis:**

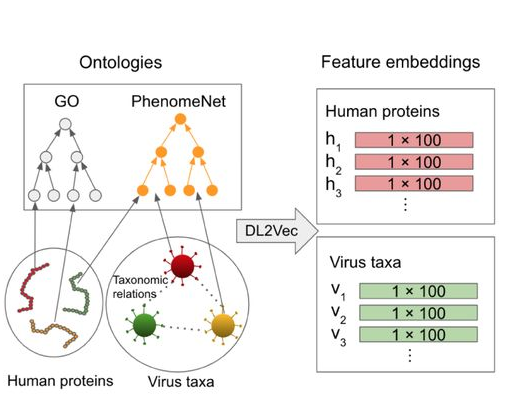
The second method is based on phenotype analysis. We compare phenotypes from the pathogens with known phenotypes associated with genes and proteins from databases containing information from human and species.

Here we perform different steps which are described below in order to generate a corpus which can be used to train a Word2Vec network.

**Description of the process:**

To generate feature embeddings, we use DL2Vec (**[Chen](https://www.biorxiv.org/content/10.1101/2020.04.22.055095v1.full" \l "ref-6) *[et al](https://www.biorxiv.org/content/10.1101/2020.04.22.055095v1.full" \l "ref-6)*[., 2020](https://www.biorxiv.org/content/10.1101/2020.04.22.055095v1.full" \l "ref-6)**), a recent method for learning features for entities (in our case, the human proteins and viruses) from their associations to ontology classes. DL2Vec first converts the ontologies and entity associations into a graph, with the classes and entities as the nodes and the associations and ontology axioms as the edges. Then several random walks are performed, starting from the entities over to the ontology graph and thereby generating a corpus of walks in the form of sentences capturing the graph neighborhoods and thereby the ontology axioms.

Following the construction of such sentences, a Word2vec skipgram model (**[Mikolov](https://www.biorxiv.org/content/10.1101/2020.04.22.055095v1.full" \l "ref-38) *[et al](https://www.biorxiv.org/content/10.1101/2020.04.22.055095v1.full" \l "ref-38)*[., 2013](https://www.biorxiv.org/content/10.1101/2020.04.22.055095v1.full" \l "ref-38)**) is used to learn an embedding for each entity by learning from the corpus. The resulting embedding is a vector representation of an entity capturing its co-occurrence relations with other entities within the graph generated by DL2Vec.



1. Creation of OWL file related to the background knowledge

Using groovy we create a background knowledge file for human/Species and a background knowledge file for the pathogens based on the database PhenomeNET Ontology and PathoPhenoDB

1. Generation of association file:

Then comes the creation of association files. In this step we create dictionaries from the different databases used as input and process these dictionaries to get association between Genes and phenotypes for the species/human and association between pathogens and phenotypes.

1. Generation of the Corpus

Using the different association file, we use a graph method and random walk and generate automatically a corpus for the pathogens and a corpus for the human/species. These 2 corpus are then merged into one global corpus.

1. Training the Word2Vec neural networks.

The global corpus is then used to train a word2vect neural network.

**Description of the word2vec neural network**

Word2Vec is one of the most popular technique to learn word embeddings using shallow neural network. It was developed by [Tomas Mikolov in 2013 at Google](https://arxiv.org/pdf/1310.4546.pdf).

Word embedding is capable of capturing context of a word in a document, semantic and syntactic similarity, relation with other words, etc. Word embedding are vector representations of a particular word.

Let’s consider the following similar sentences: *Have a good day* and *Have a great day.* They hardly have different meanings. If we construct an exhaustive vocabulary (let’s call it V), it would have

V = {Have, a, good, great, day}.

Now, let's create a one-hot encoded vector for each of these words in V. Length of our one-hot encoded vector would be equal to the size of V (=5). We would have a vector of zeros except for the element at the index representing the corresponding word in the vocabulary. That particular element would be one.

Have = [1,0,0,0,0]`; a=[0,1,0,0,0]` ; good=[0,0,1,0,0]` ; great=[0,0,0,1,0]` ; day=[0,0,0,0,1]` (` represents transpose)

If we try to visualize these encodings, we can think of a 5 dimensional space, where each word occupies one of the dimensions and has nothing to do with the rest (no projection along the other dimensions). This means ‘good’ and ‘great’ are as different as ‘day’ and ‘have’, which is not true.

Our objective is to have words with similar context occupy close spatial positions. Mathematically, the cosine of the angle between such vectors should be close to 1, i.e. angle close to 0.

Here comes the idea of generating *distributed representations*. Intuitively, we introduce some *dependence* of one word on the other words. The words in context of this word would get a greater share of this *dependence.* In one hot encoding representations, all the words are *independent* of each other*,* as mentioned earlier.

**How does Word2Vec work?**

Word2Vec is a method to construct such an embedding. It can be obtained using two methods (both involving Neural Networks): Skip Gram and Common Bag Of Words (CBOW)

***CBOW Model:*** This method takes the context of each word as the input and tries to predict the word corresponding to the context. Consider our example: *Have a great day.*

Let the input to the Neural Network be the word, *great.* Notice that here we are trying to predict a target word (*d*ay*)* using a single context input word *great.* More specifically, we use the one hot encoding of the input word and measure the output error compared to one hot encoding of the target word (*d*ay).In the process of predicting the target word, we learn the vector representation of the target word.

The input or the context word is a one hot encoded vector of size V. The hidden layer contains N neurons and the output is again a V length vector with the elements being the softmax values.

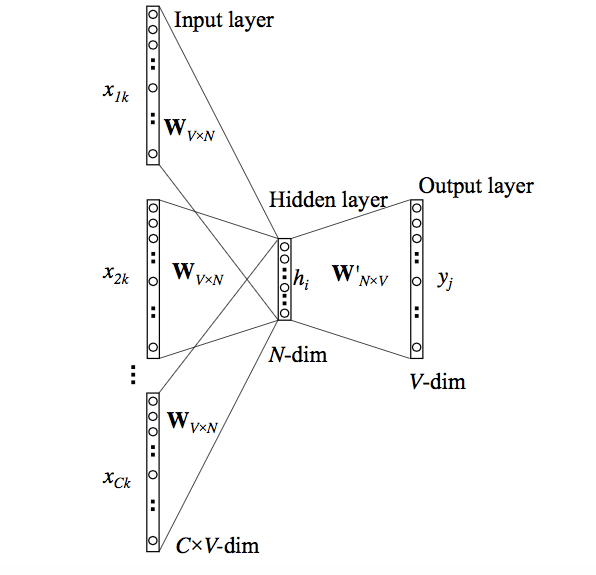
Let’s get the terms in the picture right:

*- Wvn is the weight matrix that maps the input x to the hidden layer (V\*N dimensional matrix)*

-*W`nv is the weight matrix that maps the hidden layer outputs to the final output layer (N\*V dimensional matrix)*

The hidden layer neurons just copy the weighted sum of inputs to the next layer. There is no activation like sigmoid, tanh or ReLU. The only non-linearity is the softmax calculations in the output layer.

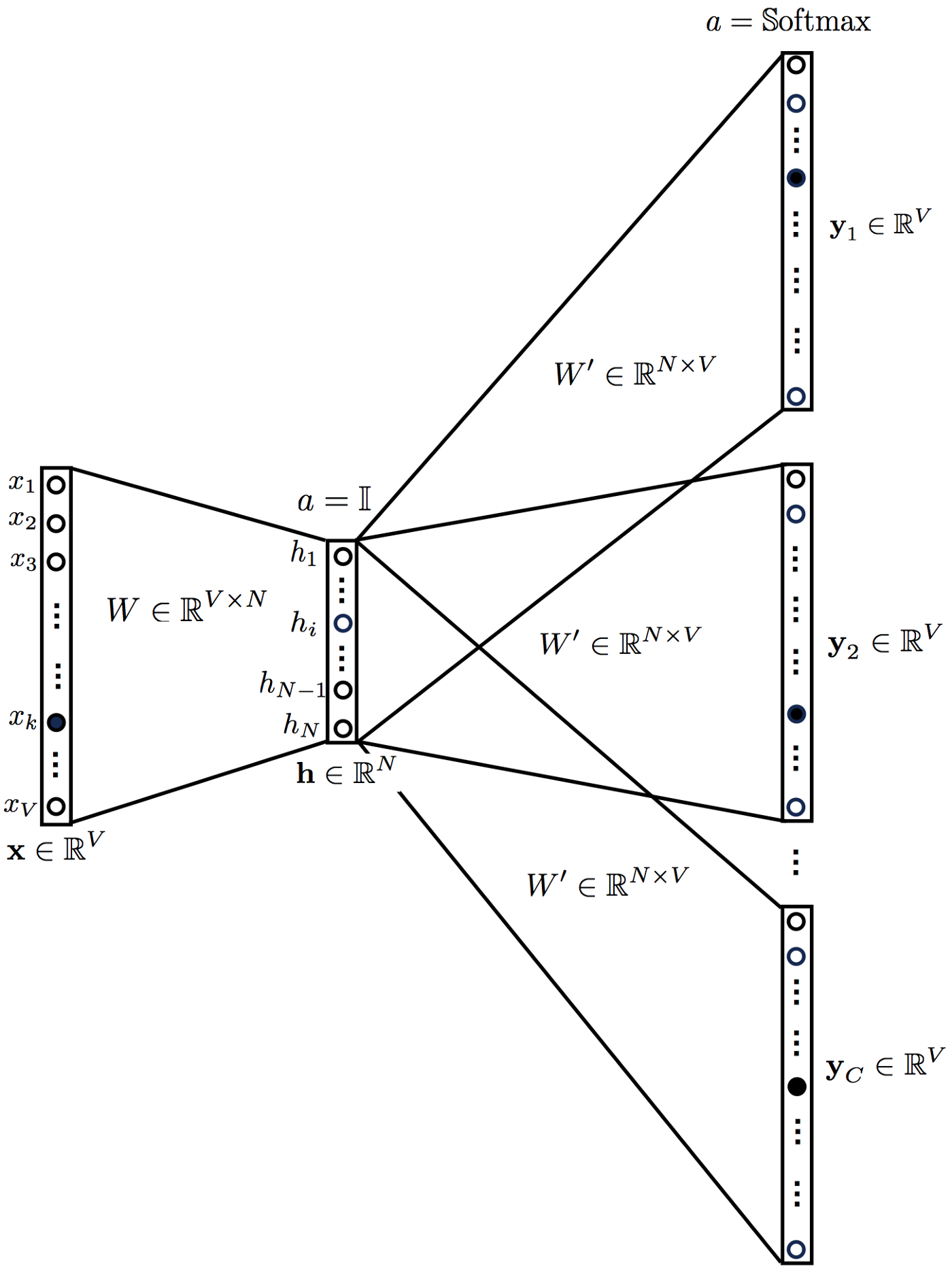
But, the above model used a single context word to predict the target. We can use multiple context words to do the same.



The above model takes C context words. When *Wvn* is used to calculate hidden layer inputs, we take an average over all these C context word inputs.

So, we have seen how word representations are generated using the context words. But there’s one more way we can do the same. We can use the target word (whose representation we want to generate) to predict the context and in the process, we produce the representations. Another variant, called Skip Gram model does this.

**Skip-Gram model:**



This looks like multiple-context CBOW model just got flipped. To some extent that is true.

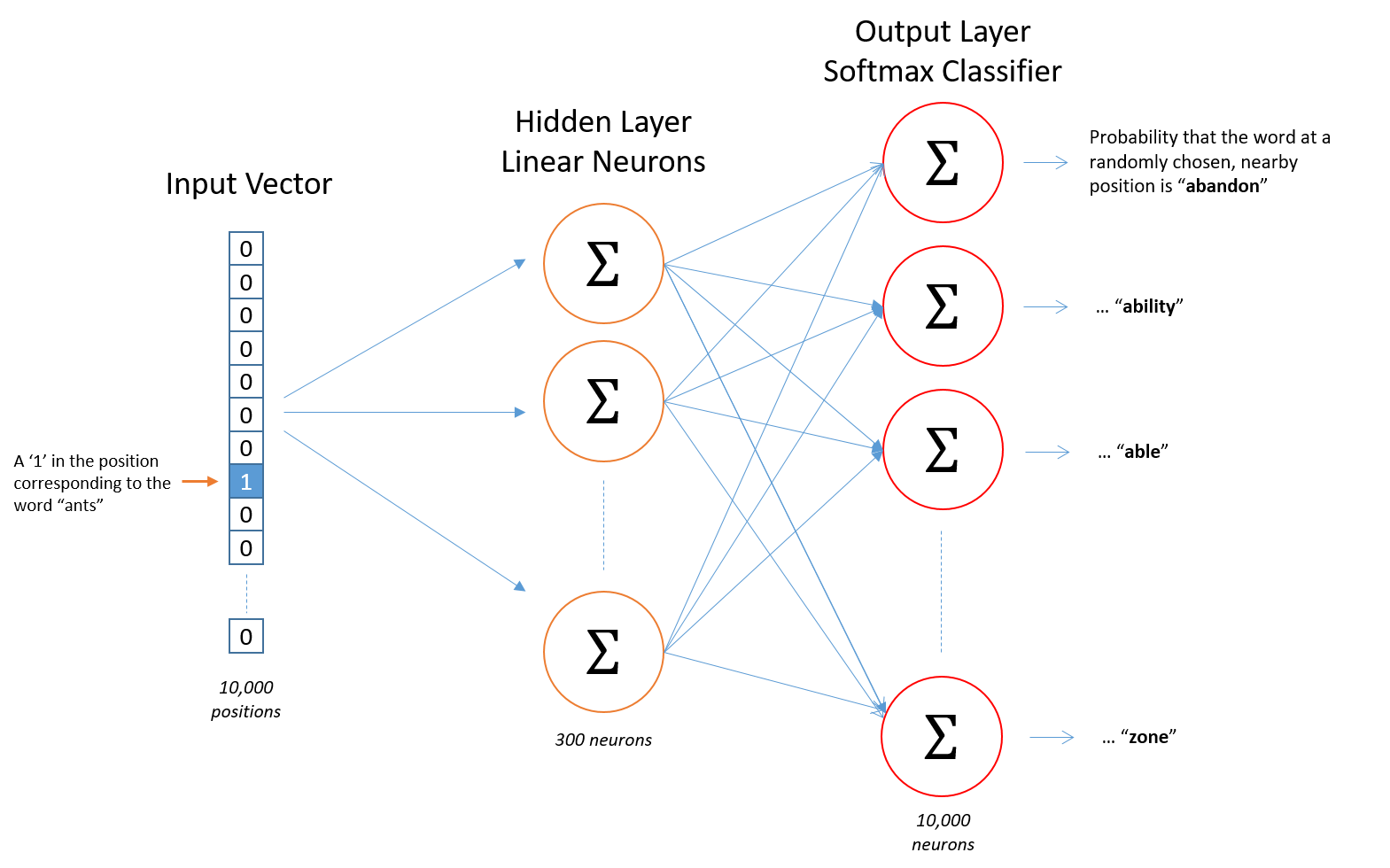
We input the target word into the network. The model outputs C probability distributions.

For each context position, we get C probability distributions of V probabilities, one for each word.

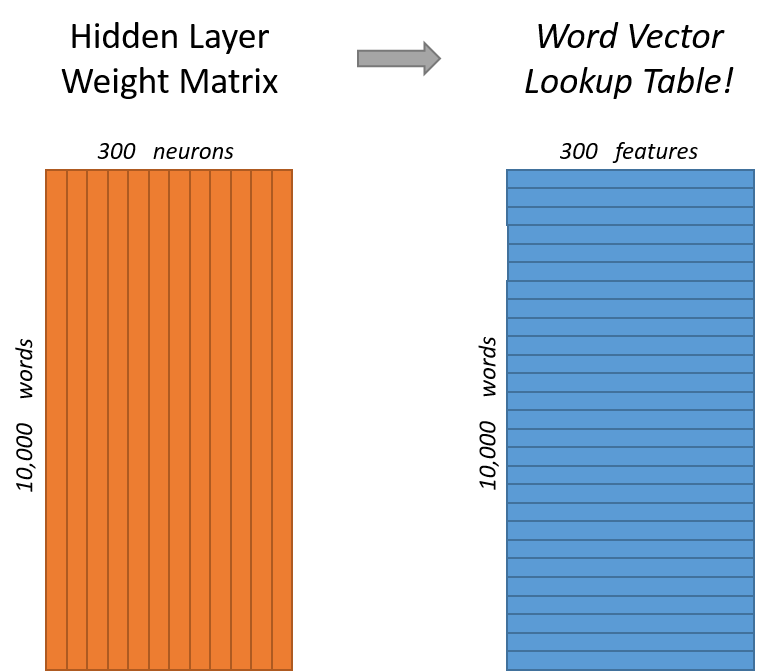
In both the cases, the network uses back-propagation to learn. Detailed math can be found [here](https://arxiv.org/pdf/1411.2738.pdf)

The above explanation is a very basic one. It just gives a high-level idea of what word embeddings are and how Word2Vec works.

Our application is based on Skip-Gram model.



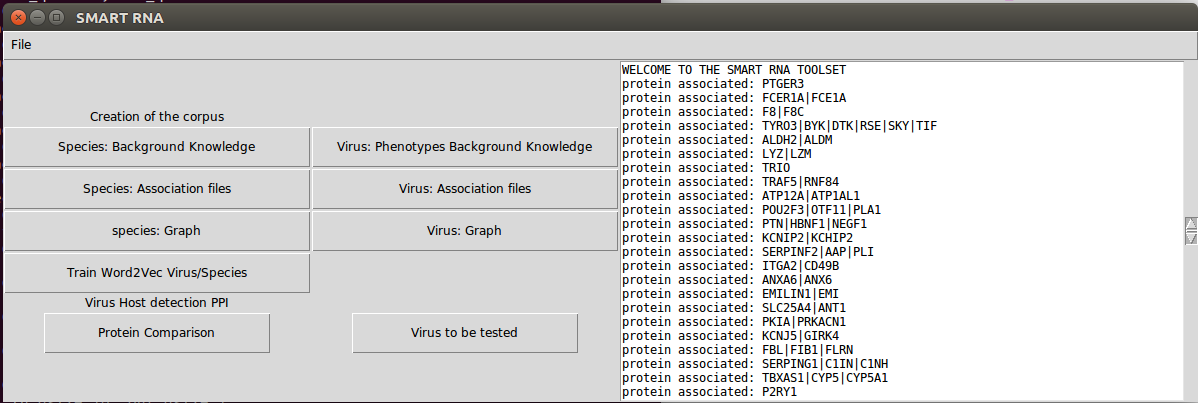
The model provides a wordemedding as output which can be represented by below figure.



1. Testing a pathogen and its phenotypes

After the training of the Wod2Vec, we can inject a vector composed of the virus and its phenotypes and we will get as output the corresponding genes, and associated proteins..

The associated proteins is then displayed by the tool as seen below:



We have now been defined by using two independent methods (protein sequence homogeneity and phenotype/oncology) potential protein interacting with the virus and genes potentially hosting the virus.

As a final step the tool is looking for correlations between the result of the two methods to output proteins which would be confirmed by the two methods.

## **3 Conclusion**

We developed a tool allowing two important function:

1. RNA sequence tracking to detect, anticipate mutation and locate origin / species / temporality of a specific RNA sequence. We used state of the art RNA alignment techniques, which we combined with a novel approach to trace the RNA sequence. This approach uses a neuromorphic neural network.
2. Identify the potential host of a virus by detecting potential genes and proteins impacted and interacting with the virus. We also use different techniques including state of the art proteins alignment method and also developed a recent method implementing powerful natural language recognition algorithms applied on virus / species phenotypes and phenotypes / Genes / proteins association.

The source code and the tool is made available for research institutes and universities and this initial project could lead to further development together with researchers to implement additional functions and propose the tool under concrete use case.

We have also identified opportunities to combine the tool to existing NGS systems to provide in real time background information of the virus RNA extracted as well detect potential error in the extracted sequence as the error rate of the NGS remains high. In addition, information extracted by the tool could be provided to the central database of the region/country to improve the knowledge of the epidemia transmission and therefore improve the control / treatment strategy of the pandemia.