**Conception d’un programme d’alignement d’embedding par programmation dynamique.**

Jean Delhomme, Tatiana Galochkina, Jean-Christophe Gelly

Université Paris Cité

# Introduction

Les algorithmes d’alignement de séquences génomiques et protéiques sont des outils indispensables de la biologie moderne. Historiquement, ces algorithmes construisent une analogie de séquences base par base ou acide aminé par acide aminé. Cette analogie représente alors une proximité évolutive ou une similarité de structure et de fonction (1). L’émergence du *deep learning* et des *language models* affine cette approche en encodant une séquence d’acides aminés en une représentation vectorielle (*embedding*) capturant les propriétés structurales et fonctionnelles de la protéine (**Figure 1**) (2).

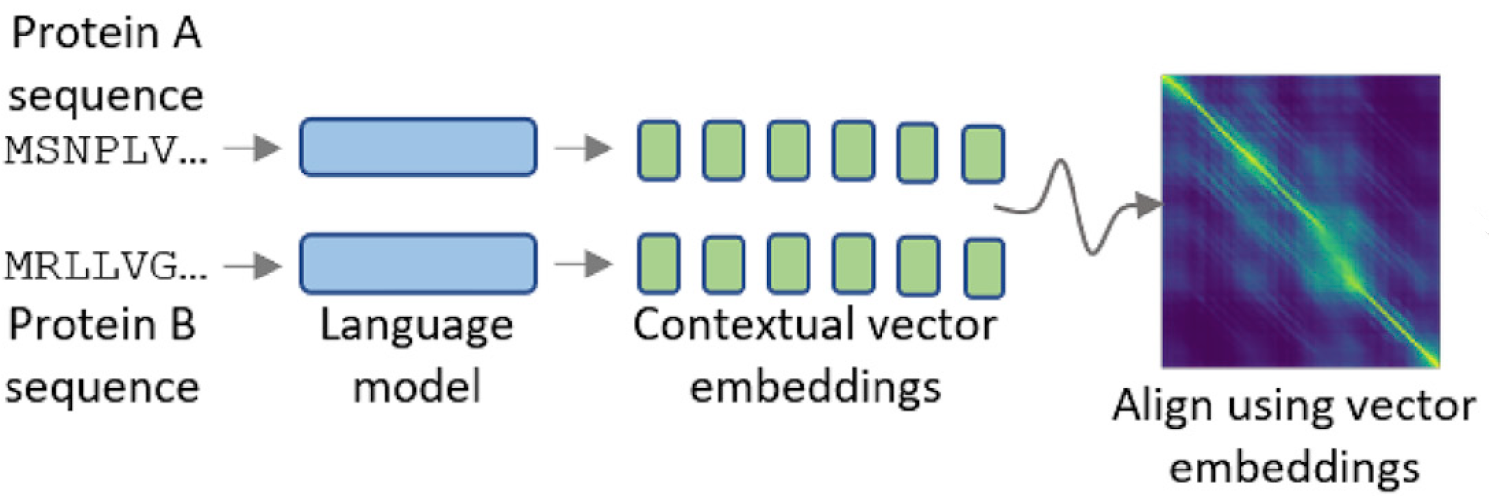


Figure 1 : Utilisation des embeddings pour un alignement de séquences. (2) Les embeddings sont générés à l’aide d’un *language model* appliqué à chaque séquence protéique. Le produit scalaire entre les vecteurs de chaque résidu de la protéine A est ensuite calculé contre les vecteurs de chaque résidu de la protéine B. La matrice obtenue est ensuite utilisée pour l’alignement et sert de matrice de similarité.

L’objectif de ce travail est de concevoir un programme dynamique permettant d’aligner des séquences représentées par un *embedding*. Le programme doit reprendre les algorithmes de Needleman et Wunsch (3), Smith et Waterman (4) et semi-global. Il sera capable de générer la matrice de similarité en calculant les produits scalaires des vecteurs d’*embeddings* des deux protéines. Il génèrera des pénalités fixes et affines de *gap*, pourra traiter des vecteurs d’*embeddings* de taille arbitraire et donnera en sortie l’alignement représenté sous forme de séquence.

# Matériel et méthodes

Le programme est exécutable depuis un terminal linux. Il utilise python 3.10.4 et numpy 1.23.1.

Le programme a recourt à des fichiers .fasta et .t5emb. Les fichiers d’embeddings ont été générés par la méthode T5 ProtTrans (5).

Nous avons utilisé l’algorithme de Needleman et Wunsch (3) pour notre alignement global et l’algorithme de Smith et Waterman pour l’alignement local (4). L’alignement semi-global a été réalisé à partir d’un algorithme de Needleman et Wunsch modifié. La **Figure 2** compare les différentes approches. Les trois algorithmes se basent sur de la programmation dynamique et ont recours à une matrice de similarité. C’est la matrice calculée à partir des produits scalaires des embeddings qui est utilisée ici.

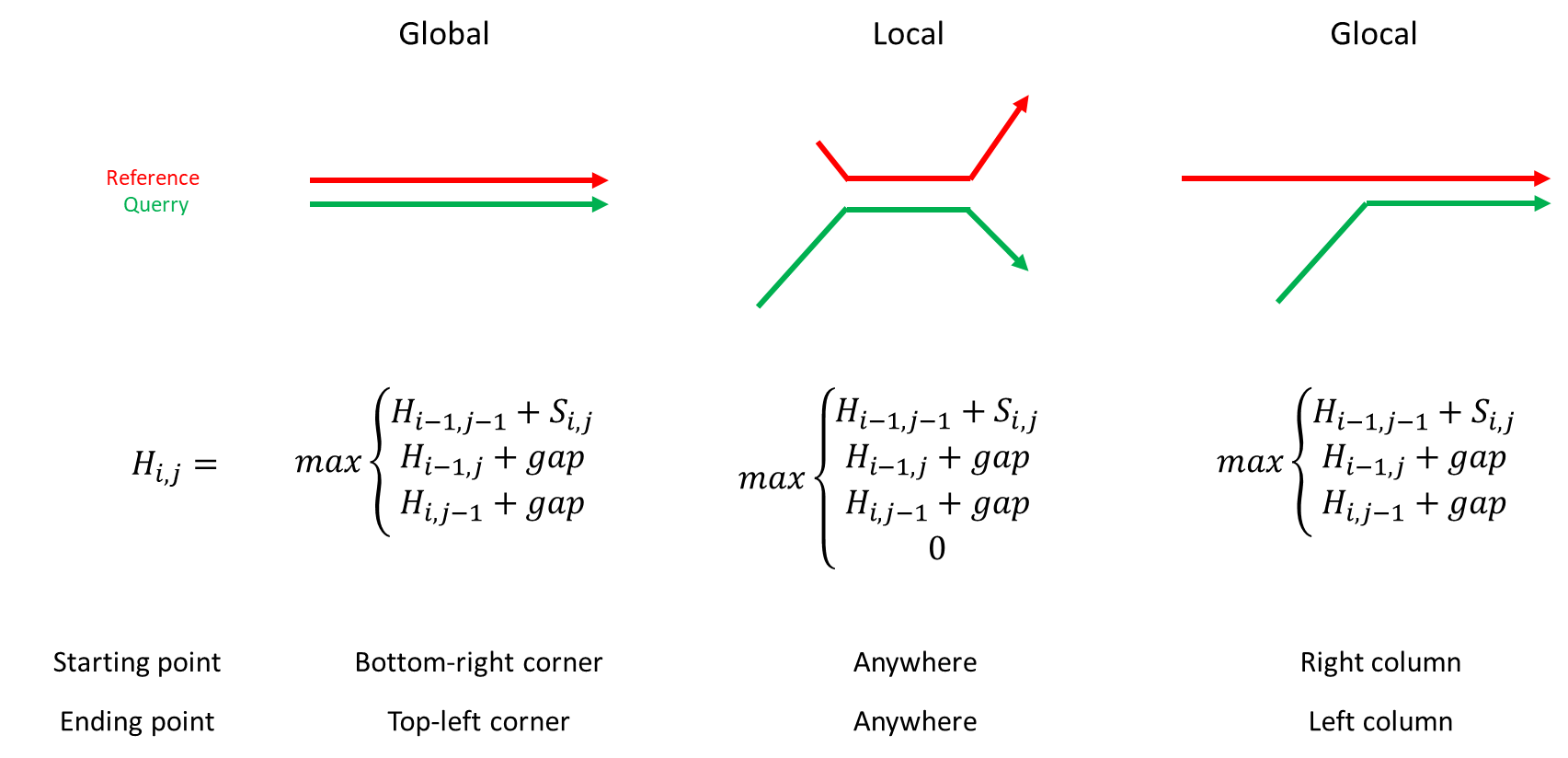


Figure 2 : Différences entre les algorithmes d’alignement global, local et semi-global. L’alignement global se fait sur l’ensemble des deux séquences. L’alignement local ne se fait que sur une région à forte homologie. L’alignement semi-global combine les deux approches, une séquence est considérée dans son intégralité tandis que l’autre non. H est la matrice de score et S la matrice de similarité. i et j correspondent aux indexes des lignes et des colonnes de ces matrices. Starting et ending point correspondent à la position des cellules utilisées pour débuter et finir le tracé de la séquence optimale.

Afin d’estimer la capacité qu’à l’algorithme de retrouver une ressemblance structurale entre deux protéines, le fichier [www.dsimb.inserm.fr/~gelly/data/TMSCORES\_HOMSTRAD.txt](http://www.dsimb.inserm.fr/~gelly/data/TMSCORES_HOMSTRAD.txt) est utilisé comme référence. Ce fichier présente ne colonne 1 l’identifiant de la première protéine, en colonne 2, l’identifiant de la seconde protéine. Les colonnes 7 et 8 correspondent aux TMscores mesurés. Un TMscore compris entre 0.5 et 1 indique la présence d’une ressemblance structurale entre les protéines.

# Résultats

**Cas d’alignement optimal**

L’algorithme est d’abord testé dans un cas optimal d’alignement. Ce test permet de s’assurer de la cohérence des résultats obtenus. Nous effectuons ce test sur la protéine 6PF2K\_1bif et sur les trois méthodes d’alignement. Les résultats obtenus sont identiques entre les trois méthodes. La **Figure 3** présente l’alignement obtenu. L’algorithme affiche également le score d’alignement de la séquence.



Figure 3 : Cas d’alignement optimal. La protéine 6PF2K\_1bif est alignée sur elle-même en suivant les algorithmes d’alignement global, local et semi-global. Les résultats des trois alignements sont identiques.

Le résultat de ce test est satisfaisant. Les deux protéines sont parfaitement alignées ce qui correspond à un TMscore de 1. Notons également que le score d’alignement obtenu de 6595.005 correspond au score obtenu dans le fichier de référence TMSCORES\_HOMSTRAD.txt ce qui nous conforte dans le bon fonctionnement de l’algorithme.

**Cas d’alignement mauvais**

Le second alignement est lancé sur deux protéines dont le TMscore est faible. Au contraire du cas d’alignement optimal, ce test permet d’observer le comportement de l’algorithme dans le cas de protéines peu homologues. L’alignement est effectué sur les protéines 6PF2K\_1bif et 7kD\_DNA\_binding\_1azpa. Les résultats obtenus sont identiques entre les trois méthodes. La **Figure 4** présente l’alignement obtenu. L’algorithme affiche également le score d’alignement de la séquence.



Figure 4 : Cas d’alignement mauvais. Les protéines 6PF2K\_1bif et 7kD\_DNA\_binding\_1azpa sont alignés en suivant les algorithmes d’alignement global, local et semi-global. Les résultats des trois alignements sont identiques.

# Discussion

# Bibliographie

1. S. Altschul et al. (2017) Handbook of Discrete and Combinatorial Mathematics. 2nd edition, Chapter 20.1 Sequence
2. T. Bepler et al. (2021) Learning the protein language: Evolution, structure, and function.
3. Needleman, Saul B. & Wunsch, Christian D. (1970) A general method applicable to the search for similarities in the amino acid sequence of two proteins.
4. Smith, Temple F. & Waterman, Michael S. (1981) Identification of Common Molecular Subsequences.
5. [A Elnaggar](https://scholar.google.fr/citations?user=fadCsRsAAAAJ&hl=fr&oi=sra) et al. (2020) ProtTrans: Towards Cracking the Language of Life's Code Through Self-Supervised Deep Learning and High Performance Computing

ANNEXES

(4 pages max)

Structure des programmes réalisés

Exemple d’utilisation

Difficultés rencontrées

python3 main.py 6PF2K\_1bif.t5emb 6PF2K\_1BIF.fasta 7kD\_DNA\_binding\_1azpa.t5emb 7KD\_DNA\_BINDING\_1AZPA.fasta nw

**NOTICE INFORMATIVE**

Le but des projets est de réaliser des programmes en utilisant des méthodes classiques de bio-informatique.

Nous souhaitons également un petit rapport sur ce projet 4 pages max figures incluses pour le rapport scientifique + 4 pages maximum pour les annexes relatives à la programmation proprement dite).

Vous pourrez détailler en annexe les difficultés que vous avez eu. Pour les projets les plus « scientifiques », le rapport comportera une petite introduction sur le but du travail, le cadre théorique, le matériel et méthodes, les résultats et un petite discussion (similaire à un article scientifique). En annexe vous présenterez la structure des programmes réalisés. Vous présenterez également un exemple d’utilisation de chaque programme (ex : prog1.py fichier1 ).

Ce rapport vise à détailler votre démarche. Un schéma ou plusieurs schémas sont vivement conseillés afin de clarifier la mise en place de votre projet et des programmes.

Pour vous donnez une idée des éléments de notation de la programmation, voici une liste, non exhaustive, des différents critères :

- Toutes les questions du projet ont été traités

- Les programmes fonctionnent et produisent le résultat attendu

- Facilité d’utilisation des programmes

- Les programmes fonctionnent avec des arguments (ex nom de fichier, paramètres...) (modulé suggéré argparse)

- Il y a une documentation associé au programme, fonction, classes ainsi que des commentaires dans le code (docstring, PEP8)

- Une fonction "help" a été implémentée pour chacun des programmes

- Concision du code

- Clarté du code

- Efficacité du code (rapidité d'exécution)

- Création et utilisation de modules

- Programmation orienté objet

Le rapport ainsi que les programmes devront "compiler" sous environnement Linux standard et devront contenir des fichiers exemples ainsi que des scénarios d'utilisations facile à mettre en place.

Une présentation individuelle présentant votre démarche et vos résultats aura lieu.

Vous préparez l’ordre de passage. La présentation sera de 7 minutes + 7 minutes de questions.

**Sujet :** **Conception d’un programme d’alignement d’*embedding* par programmation dynamique.**

OBJECTIF : réaliser un programme permettant d’aligner des séquences représentées par un embedding.

A notre disposition, on a trois ressources :

* Les données des embeddings
* Les données des séquences FASTA
* La ressemblance entre toutes les protéines (pour avoir une idée des scores que l’on devrait obtenir)

Le programme doit aligner des séquences représentées par des embeddings en reprenant les algorithmes de :

* Needleman et Wunsch
* Smith et Waterman
* Semi-global (Glocal ou Gloloc) (global sur une séquence et global sur l’autre)

Le programme :

* Calcul un score de match par le dot product (ou la correlation) entre les vecteurs d’embedding d’une position de la protéine 1 et une position de la protéine 2.
* Génère des pénalités fixes et affine des brêches (gap)
* Donne en sortie l’alignement représenté sous forme de séquence
* Permet de gérer des embedings de taille arbitraire (pas uniquement 1024)

**Article 1 : Learning the protein language : Evolution, structure, and function, 2021**

* Discussion autours des avancées sur la modélisation protéique et la prédiction de fonction. Les modèles actuels pourraient être enrichis avec les connaissances biologiques et inclure les connaissances structurales dans les représentations.

Un *language model* est une approche de *machine-learning* qui permet d’encoder une séquence d’acides aminés en une représentation vectorielle qui capture les propriétés structurales et fonctionnelles de la protéine.

* Discussion autour des avancées du language model et de ses applications en prédiction de structure.
* Comment ses modèles peuvent être enrichis des connaissances biologiques déjà existantes et comment enrichir les reprséentations existantes avec le language model ?

Introcution :

Les approches basées sur le comportement physique des atomes sont intéressantes car basées sur une compréhension fondamentale des molécules (minimisation de l’énergie libre) et sont de fait très efficaces (Rosetta Tool et GROMACS). Cependant, ces approches sont basées sur des modèles énergétiques simplifiés, sont très coûteuses en temps de calcul et ont une précision limitée.

Les algorithmes de détermination de structure sont aujourd’hui largement basés sur des modèles statistiques et de deeplearning reflétant les processus d’évolution (AlphaFold2). Ses apporches se basent sur de grandes quantités de données de séquences suffisamment similaires pour se replier d’une façon proche. Mais impossible de travailler sur des larges bases de données comprenant des protéines éloignées du point de vue évolutif. Si bien que cette méthode devient difficilement applicable avec l’augmentation des informations disponibles.

Les language models représentent une approche puissante pour déterminer des structures dans le cadre où de très grandes quantités de données sont disponibles. Le language model attribut une probabilité à chaque séquence protéique. Cette probabilité reflète les « chances » que cette séquence existe ce qui a des applications directes dans l’études des mutations. Permet aussi de faire des liens entre séquences et fonctions par transfert de connaissances.Permet enfin de faire de la génération de séquence contrôlée sur des conditions structurelles ou fonctionnelles.

Avantages :

* Capturer les liens entre acides aminés
* Peut être appliqué sur une séquence protéique entière plutôt que sur une famille

Attention, potentiel des language model n’est pas encore tout à fait réalisé, car pourrait bénéficier de plus de paramètres, de plus de capacité de calcul et de plus de données. Peuvent être cependant améliorées par les connaissances biologiques antérieurs à l’aide de multi-task learning. Possible aussi de combiner l’approche physique à celle du language model.

**Article 3 : Smith et Waterman, 1981**

Alignement local, on cherche le meilleur segment aligné dans les deux séquences.

**Questions**

* C’est quoi un embedding ?
* Comment lire les fichiers d’embedding
* C’est quoi Needleman et Wunsch ?
* C’est quoi Smith et Waterman ?
* C’est quoi semi-global ?
* C’est quoi les pénalités fixes et affines ?
* Regarder les Hidden Markov Models qui sont déjà basés sur les language models
* C’est quoi le multitask learning ?
* Voir Notion
* Voir pep8 à télécharger via conda
* Comment gérer les différents algo ? -> faire choisir l’algo dans un argument de la fonction avec un des algo par défaut.
* Pour le résultat, on affiche en output les séquences fasta
* On commence par aligner deux séquences

On donne 2 fasta

Dasta devient embedding

Fait alignement

Devient fasta

(ajouter un check pour voir si taille embeding et taille sequenece sont égales)

* On remplace les matrices de scores des algo d’aligneent classique par les coefficients de correlation de pearson des vecteurs. Pour le problème des gaps, on commence par faire des pénalités fixes on parlera des pénalités affines avec Jean Christophe. Il faudrait que la pénalité de gap soit négative. Pour les gaps, on peut commencer avec un coût de 0.

Glossaire :

transfer learning. A problem in machine learning. How we can take knowledge learned from one task and apply it to solve another

related task. When the tasks are different but related, representations learned on one task can be applied to the other. For example,

representations learned from recognizing dogs could be transferred to recognizing cats. In the case of proteins and language

models, we are interested in applying knowledge gained from learning to generate sequences to predicting function. Transfer

learning could also be applied to applying representations learned from predicting structure to function or from predicting one

function to another function among other applications.

vector embedding. A term used to describe multidimensional real numbered representations of data that is usually discrete or

high dimensional, word embeddings being a classic example. Sometimes referred to as ‘‘distributed vector embeddings’’ or

‘‘manifold embeddings’’ or simply just ‘‘embeddings.’’ Low-dimensional vector representations of high dimensional data such

as images or gene expression vectors as found by methods such as t-SNE are also vector embeddings. Usually, the goal in

learning vector embeddings is to capture some semantic similarity between data as a function of similarity or distance in the vector

embedding space.

multi-task learning. A machine learning paradigm in which multiple tasks are learned simultaneously. The idea is that similarities

between tasks can lead to each task being learned better in combination rather than learning each individually. In the case of representation

learning, multi-task learning can also be useful for learning representations that encode information relevant for all

tasks. Multi-task learning allows us to use the signals encoded in other training signals as an inductive bias when learning the

goal task.

language model. Probabilistic model of whole sequences. In the case of natural language, language models typically describe the

probability of sentences or documents. In the case of proteins, they model the probability of amino acid sequences. Being simply

probabilistic models, language models can take on many specific incarnations from column frequencies in multiple sequence

alignments to Hidden Markov Models to Potts models (direct coupling analysis) to deep neural networks.

manifold embedding. A distance preserving, low dimensional embedding of the data. The goal of manifold embedding is to find

points low dimensional vectors, z1::: zn, such that the distances, dðzi ; zjÞ , are as close as possible to the distances in the original

data space, dðxi ; xjÞ, given n high dimensional data vectors, x1::: xn. t-SNE is a commonly used manifold embedding approach for

visualization of high dimensional data.

contextual vector embedding. Vector embeddings that include information about the sequence context in which a token occurs.

Encoding context into vector embeddings is important in NLP, because words can have different meanings in different contexts

(i.e. many homonyms exist). For example, in the sentences, ‘‘she tied the ribbon into a bow’’ and ‘‘she drew back the string on her

bow,’’ the word bow refers to two different objects that can only be inferred from context. In the case of proteins, this problem is

even worse, because there are only 20 (canonical) amino acids and so their ‘‘meaning’’ is highly context dependent. This is in

contrast to typical vector embedding methods that learn a single vector embedding per token regardless of context.

autoregressive [language model]. Language models that factorize the probability of a sequence into a product of conditional

probabilities in which the probability of each token is conditioned on the preceding tokens, pðx1::: xLÞ = QL

i = 1 pðxi jx1:::xi\_1Þ. Examples

of autoregressive language models include k-mer (AKA n-gram) models, Hidden Markov Models, and typical autoregressive

recurrent neural network or generative transformer language models. These models are called autoregressive because they

model the probability of one token after another in order.

1-hot [embedding]. Vector representation of a discrete variable commonly used for discrete values that have no meaningful

ordering. Each token is transformed into a V-dimensional zero vector, where V is the size of the vocabulary (the number of unique

tokens, e.g., 20, 21, or 26 for amino acids depending on inclusion of missing and non-canonical amino acid tokens), except for the

index representing the token, which is set to one.