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 sont des outils indispensables de la biologie moderne. Ils parcourent des séquences résidu par résidu à la recherche d'analogies indiquant 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 les séquences d'acides aminés en une représentation vectorielle (*embedding*) capturant les propriétés structurales et fonctionnelles de la protéine (**Figure 1**) [(2)].

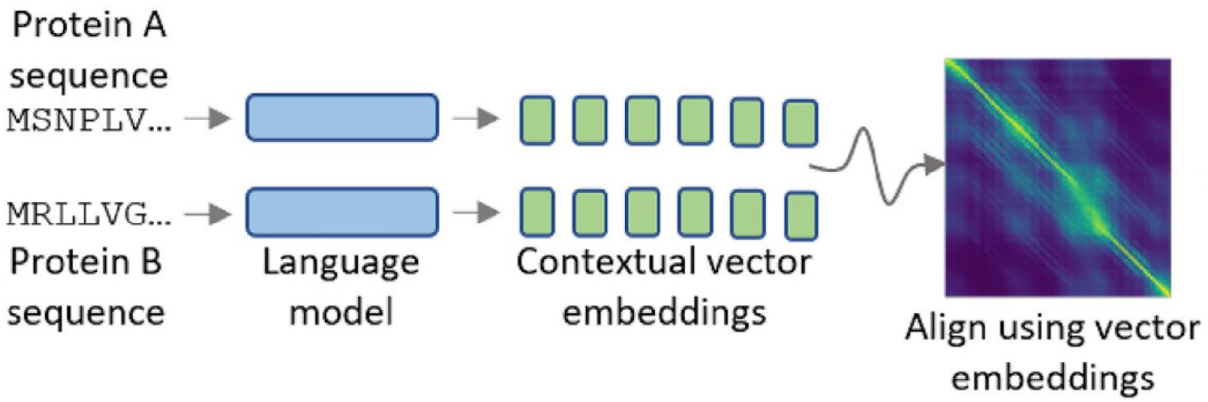


Figure 1 : Utilisation des *embeddings* pour un alignement de séquences. [(2)] Les *embeddings* sont générés à l'aide d'un *language model* appliqué à chaque séquence protéique. Le produit scalaire entre les vecteurs de chaque résidu de la protéine A est ensuite calculé contre les vecteurs de chaque 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 des *embeddings*. 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)].

L'algorithme de Needleman et Wunsch [(3)] a été utilisé pour l'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 dans ce but.

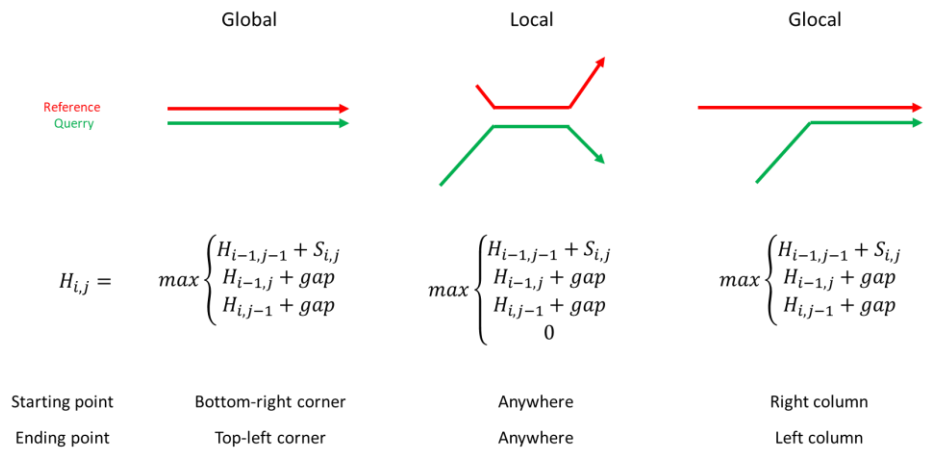


Figure 2 : Différences entre les algorithmes d'alignement global, local et semi-global. L'alignement global se fait sur l'ensemble des deux séquences. L'alignement local ne se fait que sur une région à forte homologie. L'alignement semi-global combine les deux approches, une séquence est considérée dans 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 point* 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 de l'algorithme à identifier une analogie structurale entre deux protéines, le fichier www.dsimb.inserm.fr/~gelly/data/TMSCORES_HOMSTRAD.txt est utilisé comme référence. Ce fichier présente en colonne 1, l'identifiant de la première protéine et 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 une analogie entre les deux protéines.

Résultats

Cas d'alignement optimal

L'algorithme est d'abord testé dans un cas optimal d'alignement, celui de la protéine 6PF2K_1bif sur elle-même. Ce test permet de s'assurer de la cohérence des résultats obtenus. Les trois méthodes d'alignement sont testées. Les résultats obtenus sont identiques entre les trois méthodes et sont présentés **Figure 3**. L'algorithme affiche également le score d'alignement de la séquence.

```
Global alignment of 6PF2K_1bif and 6PF2K_1bif
Alignment_score = 6595.005491138418
6PF2K_1bif
CPTLIVMVGLPARGKTYISKKLTRYLNFIGVPTREFNVGQYRRDMVKTYKSFEFFLPDNEEGL
CPTLIVMVGLPARGKTYISKKLTRYLNFIGVPTREFNVGQYRRDMVKTYKSFEFFLPDNEEGL
6PF2K_1bif
6PF2K_1bif
KIRKQCALAALNDVRKFLSEEGGHVAVFDATNTTRERRAMIFNFGEQNGYKTFFVESICVDPE
KIRKQCALAALNDVRKFLSEEGGHVAVFDATNTTRERRAMIFNFGEQNGYKTFFVESICVDPE
6PF2K_1bif
6PF2K_1bif
VIAANIVQVKLGSPDYVNRDSDEATEDFMRRIECYENSYESLDEEQDRDLSYIKIMDVGQSYV
VIAANIVQVKLGSPDYVNRDSDEATEDFMRRIECYENSYESLDEEQDRDLSYIKIMDVGQSYV
6PF2K_1bif
6PF2K_1bif
VNRVADHIQSRIVYYLMNIHVTPR
VNRVADHIQSRIVYYLMNIHVTPR
6PF2K_1bif
```

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 (6595.005) correspond exactement au score obtenu dans le fichier de référence TMSCORES_HOMSTRAD.txt. Cet élément appuie l'idée que l'algorithme est fonctionnel.

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 un cas de faible homologie. L'alignement est effectué sur les protéines 6PF2K_1bif[6] et 7kD_DNA_binding_1azpa[7]. Leurs grandes différences structurales sont appréciables **Figure 4.A.** Les résultats obtenus sont identiques entre les trois méthodes et sont présentés **Figure 4.B**. L'algorithme affiche également le score d'alignement de la séquence.

A.

B.

```
Global alignment of 6PF2K_1bif and 7kD_DNA_binding_1azpa
Alignment_score = 229.09141635142177
6PF2K_1bif
CPTLIV--MVGL-PA--RGKTYISKKLTRYLNFIGVPTREFNVGQ-YRRDM-VKTYKSFEFFLPDNEE
MVKV--KF--K-Y--KGEE-------------------KEVDTSKI-KKVWR----------------
7kD_DNA_binding_1azpa
6PF2K_1bif
GLKIRKQCALAALNDVRKFLSEEGGHVAVFDATNTTRER-RAM-I-FNF-G--E-QN-GYKTFFVEIC
-------------------------------------V-G-K-MVSF--T-YD-D--NGK-TG-----
7kD_DNA_binding_1azpa
6PF2K_1bif
ICVDPEVIAANIVQVKLG--SPDYVNRDSDEATE-DFMRRIECYENSYESLDEEQDR-DLSYIDLSYI
-----------------RGAV-----SEKDAPKE-LLDMLARAER-----------EK----------
7kD_DNA_binding_1azpa
6PF2K_1bif
IMDVGQSYVVNRVADHIQSRIVYYLMNIHVTPR
----------------------------K
7kD_DNA_binding_1azpa
```

Figure 4 : Cas d'alignement mauvais. (A) Structure des deux protéines. En haut 1bif, en bas 1azpa. (B) 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.

Là aussi, le résultat du test est satisfaisant. Les deux protéines sont alignées mais avec un nombre de *gap*s très important. Le score d'alignement est logiquement bien plus faible que précédemment (229.09). Ces observations laissent à penser que le TMscore est inférieur à 0.5. Le fichier TMSCORES_HOMSTRAD.txt propose effectivement un TMsocre de 0.16150. Comme précédemment, le score d'alignement obtenu de 229.09 correspond au score affiché dans le fichier de référence ce qui renforce encore la robustesse de l'algorithme.

Variations sur les méthodes d'alignements

Jusque-là, les différentes méthodes d'alignement ont produit des résultats identiques. Effectuer des alignements sur d'autres protéines pourrait faire apparaître des différences. L'alignement des protéines 6PF2K_1bif et adk_2ak3a[8] en révèle quelques-unes, présentées en **Figure 5**.

Les différences observées s'expliquent par le fait que la matrice d'alignement peut posséder plusieurs maximums et que ces maximums sont utilisés comme point de départ dans les méthodes locales et semi global. Plusieurs alignements optimaux sont alors proposés. Les résultats obtenus avec ces deux méthodes sont identiques et chaque méthode propose deux alignements. Ces alignements ne diffèrent que par le dernier résidu. Cette différence de résultat entre alignement global, local et semi-global laisse supposer que l'algorithme fonctionne.

Enfin, les scores d'alignement sont identiques pour toutes les méthodes (2326.07) et correspondent au score du fichier TMSCORES_HOMSTRAD.txt. Ce score d'alignement, situé entre les deux précédent correspond effectivement à un TMscore médiant, à 0.618.

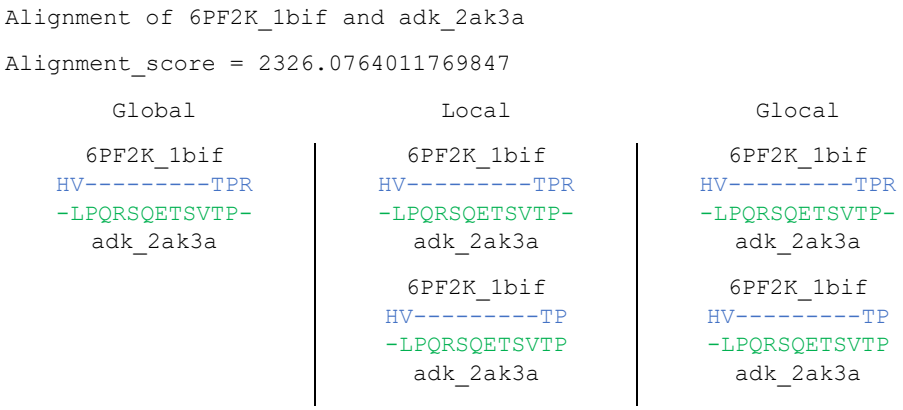


Figure 5 : Variation sur les méthodes d'alignements. Les protéines 6PF2K_1bif et 7kD_DNA_binding_1azpa sont alignés en suivant les algorithmes d'alignement global, local et semi-global. Les résultats des trois alignements sont identiques.

Discussion

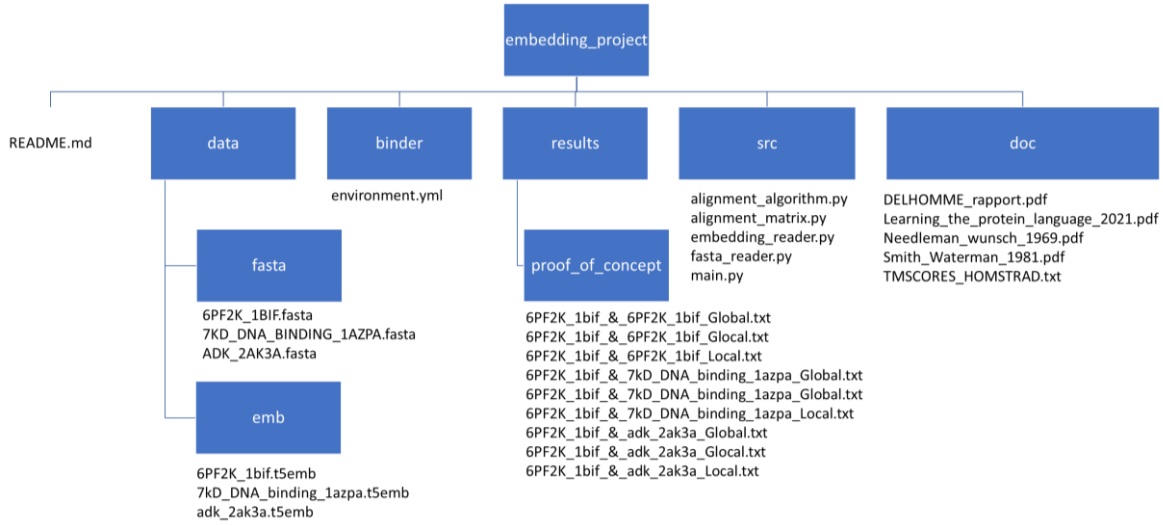
Le recours à des *embeddings* pour l'alignement de séquences s'inscrit au cœur des avancées effectuées en *deep learning* et en *language model* et promet de grandes avancées sur la qualité des prédictions de structure et de fonction par alignement. Les résultats obtenus au cours de ce travail sont encourageants sur l'utilisation des *embeddings*. En effet, les scores d'alignements obtenus évoluent dans le même sens que les TMscores de référence et sont cohérents vis-à-vis des similarités de structure observées entre les différentes protéines étudiées.

Cependant, bien qu'il ait été possible de s'assurer du bon fonctionnement de l'algorithme, les résultats obtenus entre les différentes méthodes d'alignement demeurent très similaires. La présence d'alignements alternatifs pour les algorithmes local et semi-global rassurent sur la bonne implémentation de ces méthodes mais plus de tests sont nécessaires. Utiliser le programme sur d'autres protéines jusqu'à obtenir des résultats différents pour chaque méthode permettrait de s'assurer du bon fonctionnement de l'ensemble des méthodes implémentées. La prise en compte des pénalités de *gap* affines pourrait également entrainer une plus grande variabilité entre les méthodes. Cette implémentation rendrait par ailleurs l'algorithme bien plus fidèle à la réalité biologique.

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 et al. (2020) ProtTrans: Towards Cracking the Language of Life's Code Through Self-Supervised Deep Learning and High Performance Computing
6. Page pdb de la 6-phosphofructo-2-kinase/fructose-2,6-bisphosphatase bifunctional enzyme complexed with atp-g-s and phosphate : https://www.rcsb.org/structure/1bif
7. Page pdb de la hyperthermophile chromosomal protein sac7d bound with kinked dna duplex : https://www.rcsb.org/structure/1azp
8. Page pdb de 2AK3 : https://www.rcsb.org/structure/2ak3

Annexe 1 : Structure du programme



Annexe 2 : Exemple d'utilisation

La procédure d'installation et d'utilisation complète est disponible dans le fichier embedding_project/README.md. Le programme main.py se lance comme suit :

python3 main.py argv[1] argv[2] argv[3] argv[4] argv[5]

Parameters :

argv[1] : str

The name of the embeding file of the first protein.

argv[2] : str

The name of the fasta file of the first protein.

argv[3] : str

The name of the embeding file of the second protein.

argv[4] : str

The name of the fasta file of the second protein.

argv[5] : str

OPTIONAL, the name of the alignment algorithm.
By default, Needleman and Wunsch.

nw : Needleman and Wunsch (global).

sw : Smith and Waterman (local).

gl : Glocal.

Returns :

file.txt

A text file containing the alignment result. The file is created in the results repository.

La commande ci-dessous est un exemple d'execution :

python3 main.py 6PF2K_1bif.t5emb 6PF2K_1BIF.fasta adk_2ak3a.t5emb ADK_2AK3A.fasta nw

Annexe 3 : Difficultés rencontrées

Ce projet a représenté pour moi un défi pour plusieurs raisons :

- Ma dernière expérience avec la programmation python date d'une dizaine de mois. C'est un langage que j'ai peu utilisé et avec lequel je ne suis pas très à l'aise.
- Je n'avais aucune expérience en bash, conda, github et marckdown ni en bonnes pratiques de programmations.

Malgré ces difficultés, je suis très heureux d'être parvenu à :

- Produire un programme fonctionnel en python.
- Découper mon code en fonctions et en modules que je suis parvenu à appeler depuis un script principal.
- Lancer mon script python depuis un terminal linux et pouvoir utiliser des arguments depuis ce terminal à l'aide de la méthode sys.argv.
- Construire une structure de programme cohérente.
- Rédiger un fichier README et utiliser pour cela markdown.

Il y a encore quelques éléments que j'aurais aimé pouvoir implémenter dans le cadre de ce projet :

- Intimidé par Github, je n'ai pas tout de suite pris l'habitude de *commit* et de *push* mon travail régulièrement. J'ai fini par la faire en cours de route mais je sens que je n'utilise pas l'outil à son plein potentiel.
- J'ai essayé d'implémenter une fonction *help* à mon programme principal en utilisant le module *argparse* sans succès.
- Je n'ai pas réussi à inclure les pénalités de *gap* affines à mon programme. J'aurais eu besoin de plus de temps pour l'intégrer de façon fonctionnelle.
- La programmation orientée objet me semble encore hors de portée. J'ai encore des difficultés à voir comment l'intégrer efficacement à mon code.