PARTIE PRODUCTION DONNÉES TRAITEMENT CORPUS

INTRO

Ce projet vise à analyser un corpus multilingue structuré en fichiers HTML, afin d'en extraire automatiquement des entités nommées (principalement des localisations), de procéder à leur lemmatisation linguistique, puis de les regrouper par similarité à l'aide d'un algorithme de clustering. L'application met en œuvre un pipeline modulaire, où chaque étape est déléguée à un script spécifique, facilitant la maintenance et la réutilisation dans différents contextes.

Le traitement s'articule en plusieurs phases successives : chargement des données, traitement linguistique avec spaCy, vectorisation des lemmes par n-grammes, calcul des similarités cosinus, application d'un algorithme de clustering automatique, puis export des résultats au format JSON.

Étapes de l'app

Lecture des données HTML

Les fichiers HTML sont extraits récursivement à partir d'un répertoire multilingue structuré par langue. La lecture s'appuie sur la bibliothèque BeautifulSoup, qui permet de nettoyer le contenu HTML en extrayant uniquement le texte brut. Chaque langue est représentée comme une clé dans un dictionnaire Python, pointant vers la liste de textes associés.

Ce mode de structuration permet de regrouper les fichiers par langue dès l'amont du traitement, assurant une meilleure clarté et une organisation adaptée aux traitements par lot.

Traitement linguistique

Chaque texte est ensuite analysé à l'aide de la bibliothèque spaCy. Les modèles linguistiques sont chargés dynamiquement en fonction de la langue, avec une gestion de secours via un modèle multilingue léger en cas d'absence de modèle dédié.

le traitement linguistique comprend plusieurs étapes :

- la tokenisation syntaxique,

- la suppression des noms propres et des mots outils,

- l'identification des entités nommées (NER), avec un focus sur les entités de type LOC (lieux)

- la lemmatisation des unités lexicales restantes, avec ou sans intégration des étiquettes grammaticales.

Ce traitement permet de constituer pour chaque texte une représentation lexicale épurée, focalisée sur les entités spatiales et sur le lexique non trivial.

Vectorisation et calcul de similarité

Les lemmes collectés sont ensuite vectorisés à l'aide du CountVectorizer de scikit-learn, en utilisant par défaut une analyse fondée sur les n-grammes de caractères (bigrammes et trigrammes). Ce type de vectorisation est bien adapté au traitement linguistique, car il permet de capturer les similarités morphologiques et phonétiques entre les mots, même lorsque leur racine diffère.

Une fois les vecteurs obtenus, une matrice de distance cosinus est calculée. Cette matrice est transformée en matrice de similarité en soustrayant les distances à 1. Cette matrice symétrique servira d'entrée pour le clustering.

Clustering

Le regroupement des mots similaires est effectué à l'aide de l'algorithme Affinity Propagation. Cet algorithme présente l'avantage de ne pas nécessiter la spécification du nombre de clusters à l'avance. Il s'appuie uniquement sur la matrice de similarité entre les éléments et sélectionne automatiquement les "exemplaires" ou centroïdes représentatifs des différents groupes.

Les paramètres choisis (nombre maximal d'itérations, seuil de convergence, facteur d'amortissement) sont adaptés pour permettre une convergence stable sur des corpus de taille moyenne. Une préférence est calculée à partir de la médiane des similarités non nulles, ce qui permet à l'algorithme d'éviter les extrêmes et de s'adapter à chaque langue.

À l’issue du clustering, chaque cluster est représenté par son centroïde et la liste des lemmes qui lui sont rattachés. Ces données sont organisées dans un dictionnaire indexé par identifiant de cluster.

Sauvegarde des résultats

Les résultats sont exportés dans des fichiers JSON. Chaque fichier contient les éléments suivants :

- une indication du fichier/langue traité,

- la liste complète des lemmes utilisés pour le clustering,

- la matrice de similarité (sous forme de liste de listes),

- le dictionnaire des clusters avec, pour chaque groupe, le centroïde et la liste des membres.

Le format JSON a été retenu pour plusieurs raisons :

- il permet de structurer hiérarchiquement des objets complexes (n-uplets, matrices, dictionnaires imbriqués),

- il est lisible par un humain sans outils spécialisés,

- il est compatible avec la plupart des outils de traitement de données, que ce soit en Python, JavaScript, ou dans des environnements de visualisation.

Le fichier JSON permet également de rejouer ultérieurement les analyses sans avoir à recalculer toutes les étapes amont.

Réutilisation d’un TD antérieur

Le cœur de l’algorithme, notamment la vectorisation et le clustering, a été réutilisé à partir d’un TD antérieur. Cette réutilisation s’est justifiée par la stabilité et la robustesse du code initial, ainsi que par une volonté de gagner du temps de développement. Elle a permis de concentrer les efforts sur l’intégration linguistique (traitement HTML, modèles spaCy) et sur la structuration des données.

Le traitement a été conduit sur plusieurs langues. Les langues disposant d’un corpus suffisant ont pu être traitées normalement. Pour les langues avec un volume de texte ou un taux d’entités trop faible, l’analyse a été suspendue pour éviter les surinterprétations.

Langue	Statut	Nombre de tokens
DA	OK	867
SL	OK	1033

Perspectives

L’architecture modulaire de l’application permet d’envisager facilement des extensions futures :

- enrichissement avec des statistiques lexicales (loi de Zipf, fréquences),

- intégration de visualisations (MDS, PCA, réseaux de cooccurrence),

- traitement d’autres formats d’entrée (JSONL, XML, corpus annotés).

La séparation stricte entre les étapes (lecture, traitement, clustering, sauvegarde) permet une réutilisabilité élevée et une adaptabilité à d’autres projets en TAL multilingue.

PARTIE PROUCTION GRAPHIQUE

RESULTATS :

Les résultats obtenus pour les langues sl (slovène) et da (danois) révèlent une couverture totale : tous les tokens sont assignés à un cluster. Cela peut paraître satisfaisant à première vue, mais en réalité, la structuration obtenue reste très faible. En effet, on observe une majorité de clusters unitaires, appelés singletons : environ 80 % des clusters en sl, et 70 % en da, ne contiennent qu’un seul mot. Cela suggère soit une grande hétérogénéité du vocabulaire (notamment en morphologie ou orthographe), soit un seuil de regroupement trop strict, ce qui empêche la formation de familles lexicales significatives.

La distance moyenne entre les tokens est elle aussi élevée, surtout pour le danois (0.85), traduisant une grande dispersion dans l’espace vectoriel de similarité. On a affaire à des mots qui, globalement, ne sont pas proches les uns des autres. Pour le slovène, la distance est légèrement plus faible (0.71), mais le nombre d’intersections inter-clusters est bien plus important (plus de 86 000, contre seulement 4 800 pour le danois). Cela révèle une tendance importante à la redondance

ou au chevauchement entre clusters, ce qui peut mettre en question la robustesse du critère de regroupement utilisé.

La visualisation MDS des 5 plus grands clusters apporte une lecture immédiate, mais insuffisante. Certes, on distingue les principaux regroupements par la couleur, mais leur taille reste faible (entre 13 et 23 mots en général), et tout le reste du lexique apparaît en gris, dilué dans un fond massif peu lisible. Sans interaction (zoom, clics, filtrage dynamique), on ne peut pas explorer finement les relations entre les tokens.

Du côté des ngrammes, les projections MDS révèlent des structures typiques notamment un effet de fer à cheval ou de croissant classique en MDS lorsque les données sont linéairement corrélées. Ces projections ne permettent pas d'interpréter directement des familles lexicales ou sémantiques, mais elles donnent une bonne intuition sur la proximité morphologique entre les formes. Cela est dû à l'utilisation du CountVectorizer sur des ngrammes de caractères, combiné à la distance cosinus. Cette approche est très adaptée à l'analyse formelle (préfixes, suffixes, racines partagées), mais reste aveugle à la syntaxe, la grammaire ou le sens.

Concernant la méthode de clustering utilisée, elle n'est pas explicitement décrite ici. Mais le grand nombre de clusters (553 pour sl, 361 pour da) laisse penser qu'il s'agit d'une approche fondée sur un seuil de similarité (par exemple : seuil sur matrice de distance), plutôt que d'un algorithme classique comme k-means ou DBSCAN. Le fait que la majorité des clusters soient de taille 1 ou 2 pourrait être le symptôme d'un seuil trop élevé, qui empêche les regroupements sémantiquement viables.

Pour aller plus loin, plusieurs pistes d'amélioration sont envisageables. D'abord sur l'aspect visuel : il serait utile d'annoter les centroïdes les plus significatifs, de réduire l'emphase sur les points gris (moins d'opacité, moins de taille), et d'implémenter une version interactive via une librairie comme Plotly ou Bokeh. Ensuite, sur l'aspect analytique : on pourrait filtrer les clusters les plus petits, ou appliquer une seconde couche d'agrégation pour regrouper les familles proches. Cela améliorerait la lisibilité globale des résultats.

Enfin, du point de vue méthodologique, il serait pertinent de tester des projections non linéaires comme UMAP, qui conservent mieux les distances locales et permettent une visualisation plus fidèle des structures internes. Sur le fond, des embeddings sémantiques (comme ceux de FastText, spaCy ou même BERT) permettraient d'intégrer la dimension du sens dans les regroupements, et de sortir d'une logique purement formelle.

L'ensemble des résultats pourrait aussi être exporté dans des formats lisibles et exploitables (CSV, JSON) pour documenter les clusters les plus significatifs, ou générer automatiquement un rapport visuel (PDF ou HTML) qui synthétise les statistiques, les graphes, et les observations. Cela faciliterait leur interprétation et leur diffusion dans un cadre de recherche ou d'analyse linguistique appliquée.

TECHNIQUE :
La technique utilisée repose sur une chaîne de traitement linguistique combinant des outils d'analyse de forme et de visualisation dimensionnelle. Pour mesurer la similarité entre les mots, un vectoriseur de type CountVectorizer est appliqué sur des ngrammes de caractères (bigrams, trigrams, etc.), ce qui permet de capturer les régularités morphologiques et orthographiques. Les vecteurs obtenus sont ensuite comparés via la distance cosinus pour construire une matrice de dissimilarité. Cette matrice est projetée en deux dimensions grâce à l'algorithme de réduction MDS (Multi-Dimensional Scaling), issu de la bibliothèque scikit-learn, permettant une représentation visuelle des proximités entre tokens. Les visualisations sont réalisées avec matplotlib, en intégrant des codes couleurs et des formes pour représenter les clusters, les centroïdes, et les catégories grammaticales. Le traitement JSON et la gestion de fichiers sont assurés par les bibliothèques standards json, glob et os. L'ensemble repose sur une architecture Python simple, modulaire et facilement extensible.

Langue	Tokens totaux	Tokens clusterisés (%)	Clusters	Clusters singletons	Distance globale moy.	Intersections totales	Taille moy. cluster	Taille min / max	Pureté approx.
sl	1033	1033 (100.0%)	553	439	0.7190	86 322	1.87	1 / 20	0.019
da	867	867 (100.0%)	361	252	0.8490	4 885	2.40	1 / 23	0.027

ANALYSE GRAPH :

Analyse comparée des clusters MDS pour le danois et le slovène

Les deux graphiques ci-dessus présentent une réduction de dimension (MDS Multidimensional Scaling) appliquée aux matrices de similarité entre tokens obtenues après vectorisation par n-grammes de

caractères. L'objectif est de projeter, dans un espace bidimensionnel, la structure des similarités lexicales entre les lemmes extraits à partir des entités nommées géographiques, en particulier les toponymes. L'algorithme de clustering utilisé étant AffinityPropagation, il produit des regroupements de mots autour de centroïdes identifiés automatiquement. Ces regroupements sont ensuite projetés dans l'espace MDS, les cinq plus gros clusters étant représentés par des couleurs et des formes distinctes.

Sur le plan visuel, le danois montre une répartition plus aérée des points, avec des groupes de mots qui se différencient spatialement de manière relativement nette. Les cinq plus grands clusters identifiés occupent des zones distinctes dans l'espace, signe d'une différenciation forte entre les lemmes au regard des n-grammes qui les composent. On observe une structuration du lexique qui suggère que certaines racines ou motifs morphologiques (préfixes, suffixes, consonances internes) sont suffisamment distincts pour créer des regroupements saillants, bien séparés dans l'espace vectoriel. Cette différenciation est renforcée par une distance globale moyenne élevée (0.8490), indiquant que les mots sont globalement peu similaires entre eux. Cela s'accompagne d'un nombre modéré de clusters (361), dont une majorité de taille supérieure à 1 (taille moyenne de 2.40), et d'une pureté approximative relativement correcte pour un clustering non supervisé (0.027). Ces indicateurs convergent vers une structuration lexicale en groupes bien définis.

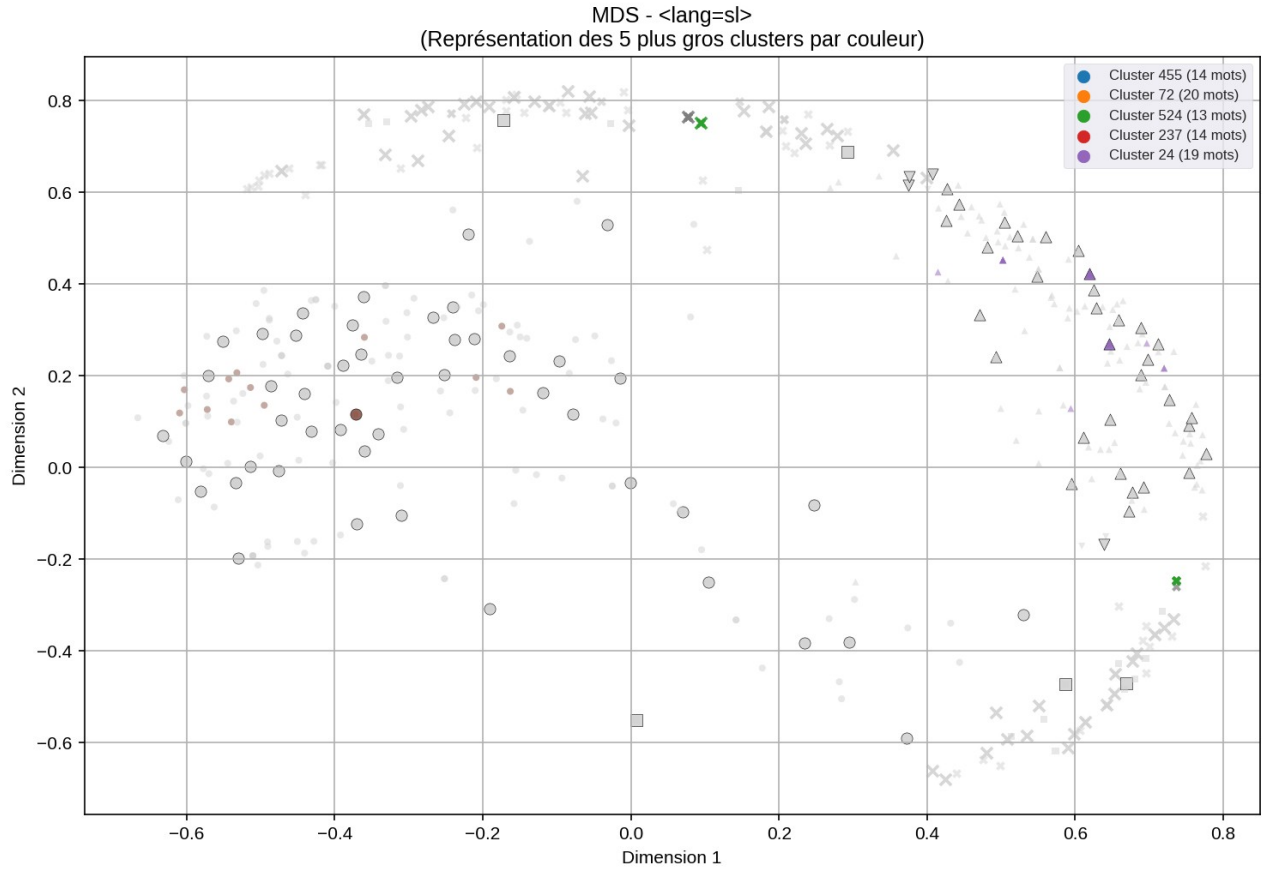
Le cas du slovène est sensiblement différent. Le nuage de points projeté dans l'espace MDS suit une trajectoire plus continue, presque en forme d'arc ou de croissant. Cette configuration est souvent associée à une organisation hiérarchique ou à une transition progressive dans les similarités lexicales. Ici, elle peut refléter un espace morphologique très dense, dans lequel les mots partagent entre eux de nombreux segments communs, mais sans former pour autant de grappes très distinctes. Cette hypothèse est soutenue par la prolifération de petits clusters (553 au total pour 1033 tokens), dont 439 sont des singletons. Cette explosion du nombre de micro-clusters rend la représentation globale plus difficile à segmenter, comme en témoigne la pureté moyenne plus faible (0.019) et la taille moyenne des clusters limitée (1.87). Visuellement, les cinq plus grands groupes sont également moins bien individualisés, avec une tendance à se recouvrir partiellement. Leur proximité spatiale peut indiquer soit une ressemblance réelle entre leurs centroïdes (formes proches phonologiquement ou morphologiquement), soit un effet d'ambiguïté dans la vectorisation n-grammique.

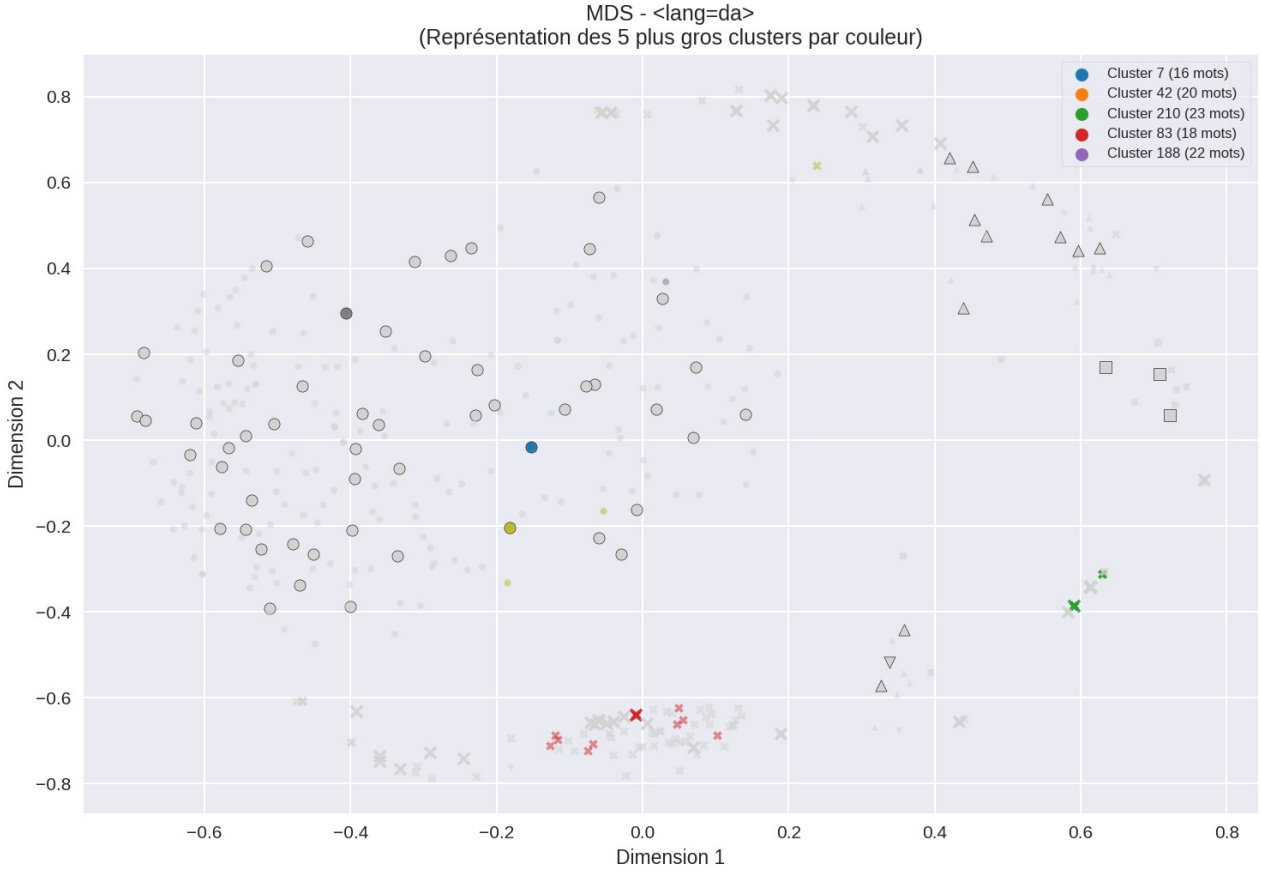
Ces différences structurelles et visuelles doivent être replacées dans le contexte morphologique propre à chaque langue. Le danois, langue germanique du nord, présente une structure morphologique relativement simple avec un lexique souvent analytique et peu de flexions visibles. Les toponymes danois tendent également à avoir une formation régulière, avec des suffixes récurrents (-havn, -borg, -sted, etc.), ce qui favorise leur regroupement par similarité de n-grammes. Ainsi, un cluster pourra par exemple regrouper des villes terminant en -havn ou en -køb*ing*, créant des unités lexicales cohérentes du point de vue des segments de caractères.

À l'inverse, le slovène est une langue slave hautement flexionnelle. Même dans le cas des entités nommées, les formes peuvent varier en fonction des cas grammaticaux (nominatif, accusatif, génitif...), ce qui introduit une dispersion formelle accrue. Un même toponyme peut apparaître sous plusieurs formes différentes dans le corpus, ce qui augmente artificiellement le nombre de lemmes à clusteriser, et tend à fragmenter l'espace lexical. De plus, les suffixes slaves sont souvent très productifs et peuvent générer des similarités partielles entre des mots qui ne sont pas sémantiquement proches. En vectorisation par n-grammes, cela entraîne une proximité élevée entre formes pourtant distinctes, ce qui biaise le clustering vers des regroupements morphologiquement arbitraires. C'est probablement ce phénomène qui explique la structure continue observée dans l'espace MDS du slovène, avec une faible séparation entre les groupes.

Le choix du n-gramme de caractères comme base de vectorisation, bien qu'efficace dans des langues peu flexionnelles, montre ici ses limites dans des langues à morphologie riche. Il ne permet pas toujours de capturer les régularités sémantiques ou syntaxiques pertinentes, et peut au contraire accentuer les effets de surface (par exemple des suffixes fréquents partagés par des entités sans lien lexical réel). Une alternative serait d'introduire une vectorisation sémantique ou morpho-syntaxique (par embeddings contextualisés ou analyse morphologique supervisée), mieux adaptée à la granularité du slovène.

Enfin, ces différences entre les deux langues révèlent aussi des implications sur le plan du traitement des corpus multilingues. Les configurations d'espace lexical observées ici soulignent l'importance de choisir des représentations adaptées aux propriétés morphologiques des langues traitées, et de calibrer les paramètres de clustering en conséquence. Dans une perspective de traitement plus large, il serait pertinent d'intégrer une étape de normalisation morphologique (lemmatisation plus robuste, désambiguïsation des cas), voire d'expérimenter une vectorisation multilingue pour harmoniser la comparaison entre langues typologiquement éloignées.


MDS - <lang=sl>
(Représentation des 5 plus gros clusters par couleur)
Cluster 455 (14 mots)
Cluster 72 (20 mots)
Cluster 524 (13 mots)
Cluster 237 (14 mots)
Cluster 24 (19 mots)
Dimension 1
Dimension 2


MDS - <lang=da>
(Représentation des 5 plus gros clusters par couleur)
Cluster 7 (16 mots)
Cluster 42 (20 mots)
Cluster 210 (23 mots)
Cluster 83 (18 mots)
Cluster 188 (22 mots)
Dimension 1
Dimension 2

Intro

Les quatre figures représentent des projections bidimensionnelles via MDS des tokens vectorisés selon des n-grammes de caractères. Elles permettent d'évaluer la distribution et la densité des lemmes selon leurs similarités internes.

Langue	N-grammes	Description de la projection
da	(2,3)	Structure légèrement ramifiée, forte densité centrale, peu de zones vides
sl	(2,3)	Structure en arc très marquée, forte continuité, peu de ruptures
da	(4,5)	Structure plus éclatée, segments distincts, zones de vide plus nombreuses
sl	(4,5)	Arc toujours présent mais plus fragmenté, apparition de sous-groupes disjoints

2. Impact du choix de la taille des n-grammes

a) (2,3) n-grammes courts

Les bigrammes et trigrammes capturent des similarités locales très fines. Ils sont sensibles aux affinités de surface : préfixes fréquents, syllabes communes, combinaisons courtes de sons ou lettres. Cela a pour effet de créer une forte densité dans l'espace vectoriel, car beaucoup de mots partagent des segments courts identiques.

Avantages : bonne couverture des similarités superficielles, utilité pour identifier des variantes morphologiques proches.

Limites : effet de bruit important ; des mots très différents peuvent être perçus comme similaires à cause de segments triviaux (ex. "ka", "ra", "li").

Dans les deux cas (da et sl), les projections (2,3) produisent des nuages très compacts. Pour le slovène, on observe une structure en fer à cheval classique en MDS, symptôme d'un espace sous-jacent fortement corrélé. Pour le danois, bien que la forme soit moins marquée, on note une densité centrale, où les points sont très proches, confirmant une accumulation de mots très similaires du point de vue n-grammique court.

b) (4,5) n-grammes longs

Les 4-grammes et 5-grammes augmentent considérablement la précision. Ils capturent des motifs plus spécifiques, souvent propres à une racine ou un segment lexical complet. Les mots sont ainsi mieux différenciés, et le modèle tend à isoler davantage les formes lexicales éloignées.

- Avantages : meilleure séparation des familles morphologiques, réduction des faux positifs de similarité.

- Limites : moins robuste pour les mots courts ; les lemmes très différents mais partageant quelques longues séquences sont survalorisés.

Le changement est notable. Pour le danois, l'espace se fragmente. On observe des zones denses et des vides marqués, ce qui indique que les mots se structurent en familles plus distinctes. Les segments visuellement séparés peuvent correspondre à des classes morphologiques différentes (ex. suffixes géographiques, composés). Pour le slovène, le fer à cheval se rompt partiellement : l'arc subsiste mais devient moins homogène, avec des "trous" et des sous-groupes qui émergent. Cela traduit une meilleure différenciation entre des formes lexicales auparavant regroupées, mais aussi un effet de perte de cohésion dans un espace morphologiquement très dense.

3. Analyse Langue et sémantique

Danois

Langue à morphologie relativement simple, le danois utilise de nombreux composés mais peu de flexions. Les toponymes et entités nommées géographiques suivent souvent des motifs réguliers, avec des suffixes récurrents. Les 2-grammes et 3-grammes capturent bien ces régularités, mais sont limités dans leur capacité à différencier les sous-classes. L'utilisation de 4-grammes et 5-grammes révèle plus clairement des distinctions sémantiques ou topologiques : les clusters peuvent correspondre à des zones géographiques spécifiques, ou à des constructions morphologiques (préfixe + racine, par exemple).

En d'autres termes, les grands n-grammes isolent des segments porteurs de sens lexical, tandis que les courts captent la texture morphophonologique.

Slovène

Langue flexionnelle avec une morphologie riche, le slovène présente de nombreuses variations formelles pour une même entité. Les formes fléchies, même lemmatisées, peuvent garder des traces de suffixation ou d'alternance vocalique. Les n-grammes courts ne permettent pas de capturer des familles sémantiques solides, car ils surinterprètent des proximités de surface. L'espace lexical devient donc excessivement compact ou lisse (structure en arc). Avec les 4-grammes et 5-grammes, l'algorithme distingue mieux des racines complètes ou des segments internes plus pertinents sémantiquement, ce qui permet un début de structuration sémantique. Cependant, l'effet de fragmentation reste fort, ce qui indique que l'empreinte morphologique du slovène est trop complexe pour une analyse reposant uniquement sur des caractères en surface.

Critère	da (2,3)	da (4,5)	sl (2,3)	sl (4,5)
Densité dans l'espace MDS	Élevée, centralisée	Moyenne, zones claires	Très élevée, structure en arc	Moyenne, arc fragmenté
Différenciation morphologique	Faible	Bonne	Faible	Meilleure mais limitée
Fragmentation des groupes	Modérée	Importante	Très faible (amas global)	Forte (sous-groupes visibles)
Pertinence sémantique	Faible (formes proches phonétiquement)	Meilleure (racines partagées)	Faible (bruit morphologique)	En progrès (début de structuration)
Stabilité des formes courtes	Bonne	Moins bonne	Moyenne	Moins bonne

Ces observations soulignent l'importance de calibrer les tailles de n-grammes en fonction :

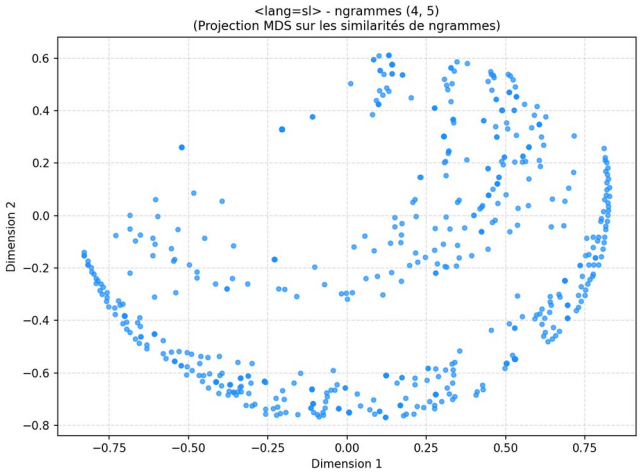
- de la morphologie de la langue (analytique vs. Flexionnelle),

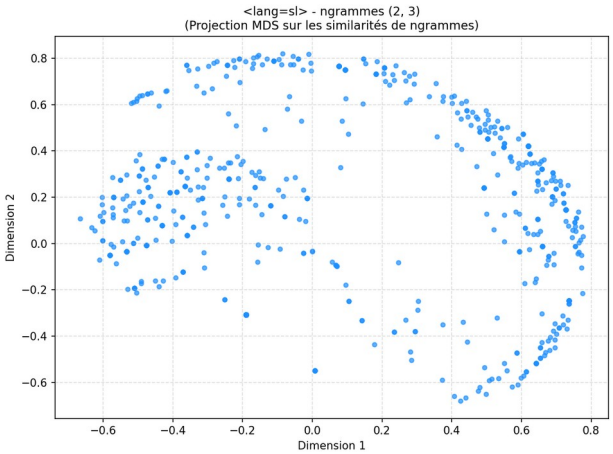
- du type de données lexicales (noms propres, composés, entités),

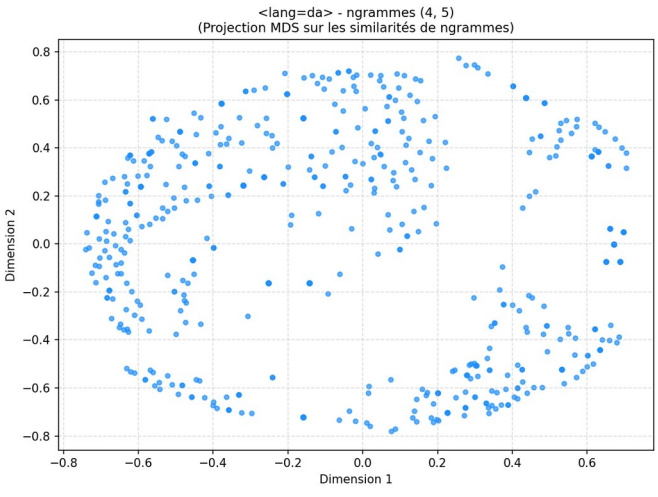
- de l'objectif d'analyse (regroupement sémantique, étude morphologique, cartographie lexicale).

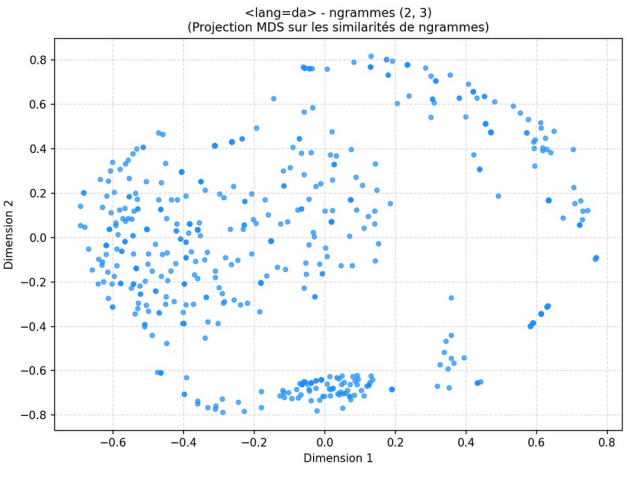
Les résultats montrent que les n-grammes courts ont tendance à favoriser une structure de similarité homogène mais bruitée, tandis que les n-grammes longs permettent une segmentation plus précise, au risque de générer une fragmentation excessive.

Dans une optique d'analyse sémantique approfondie ou de clustering thématique, il serait recommandé d'hybrider les représentations n-grammiques avec des embeddings contextuels ou de renforcer la lemmatisation par des outils morphologiques supervisés adaptés à chaque langue.









Réutilisation d’une architecture antérieure : pertinence et adaptation au multilingue

Le traitement décrit ci-dessus s’appuie en grande partie sur une architecture logicielle préexistante, initialement développée dans le cadre d’un TD consacré à l’analyse d’entités nommées sur des fichiers .bio mono-langues. Ce choix de réutilisation s’est avéré particulièrement judicieux, tant sur le plan du gain de temps que de la robustesse du traitement. L’architecture originelle était déjà organisée de façon modulaire, avec une séparation claire entre les différentes étapes du pipeline : extraction des données, traitement linguistique, vectorisation, clustering, puis visualisation.

Dans le cadre de ce projet multilingue, les modifications nécessaires ont été ciblées et limitées à la source des données (reader_html.py remplaçant reader_bio.py) et à la gestion des modèles linguistiques dans spacy_processor.py. Le reste du pipeline en particulier la vectorisation n-grammique (vectorizer.py), le clustering via Affinity Propagation (clusterer.py) et la sauvegarde des résultats en JSON (saver.py) a pu être conservé tel quel. Cela a permis une adaptation rapide à la nouvelle tâche, tout en assurant une cohérence technique et méthodologique avec les travaux antérieurs.

Le cœur du système repose sur un script central (global_launch.py) qui orchestre l’exécution des différentes étapes, ce qui facilite les tests, le déploiement et la maintenance du projet. Cette architecture a donc permis non seulement de réduire la charge de travail, mais aussi de garantir la réutilisabilité des modules dans d'autres contextes de traitement linguistique.

Enfin, la compatibilité avec le système de visualisation préexistant a été maintenue : les fichiers JSON produits sont directement exploitables par l’application de projection MDS, sans adaptation supplémentaire. Cette continuité dans le format de données confirme l’intérêt d’une normalisation en amont des sorties, et montre que des choix d’architecture bien pensés peuvent faciliter l’évolution vers des scénarios d’analyse plus complexes, notamment en traitement multilingue.