Apprentissage de composition pour la représentation des groupes nominaux

Yannis Coutouly

Juillet 2021

1 Introduction

Ce rapport rend compte du travail produit lors d'un stage de deux mois que j'ai effectué à la fin de ma L3, sous la supervision de Alexis Nasr et Carlos Ramisch. L'objectif était de construire des embeddings pour n'importe quelle séquence de mots. Pour cela on cherche des fonctions qui composent des embeddings de mots simples en embeddings de formes construites. Une forme construite est une séquence de mots que l'on regroupe en une seule unité syntaxique. Par exemple dans la phrase, "La voiture rouge roule", "La voiture rouge" aurait un embedding (propre).
Pour obtenir les embeddings de formes construites on utilise deux approches : l'approche par concaténation et l'approche compositionnelle. L'approche par concaténation vise/cherche à concaténer les formes construites dans le corpus pour produire les embeddings avec Word2Vec. Et enfin trouver une fonction qui retrouve l'embedding des formes construites à partir de ses composants. L'approche compositionnelle utilise un principe d'auto-encodeur. On code et décode nos mots dans un espace intermédiaire qui contient nos embeddings de formes construites.
La construction d'embeddings de séquence pourrait faire le lien entre deux approches de la sémantique. La sémantique distributionnelle avec les embeddings et la sémantique compositionnelle avec les fonctions de compositions.

2 Approche par concaténation

2.1 Principe général

Le principe de cette approche est d'utiliser Word2Vec pour construire les embeddings de nos formes construites. Pour cela, on modifie notre corpus avec des tirets pour que chaque forme construite correspond à un seul token. Une fois ces embeddings crées, on entraîne un MLP à retrouver les embeddings des formes construites en partant des embeddings de leurs composants.

On travail avec des formes construites qui ont la structure "N1 de N2" (Exemple : "question de pouvoir"). Dans notre corpus on va modifier "question de pouvoir" en "question-de-pouvoir". Chaque forme construite correspond à un seul token, donc Word2Vec va attribuer un embeddings à chacune d'entre-elles.

Ensuite, on entraîne un MLP à reconstruire nos formes construites à partir des mots qui les composent. On présuppose donc que le sens de notre forme construite peut être déterminé à partir du sens de ces composants, c'est l'hypothèse compositionnelle (A reformuler je pense).

La couche d'entrée de ce MLP est de taille 600 avec du Dropout à 50%. La couche Dense est de dimension 480 avec une fonction d'activation "relu", la couche de sortie est de taille 300 et utilise une fonction tangente hyperbolique. La fonction de loss est une similarité cosinus entre l'embedding produit

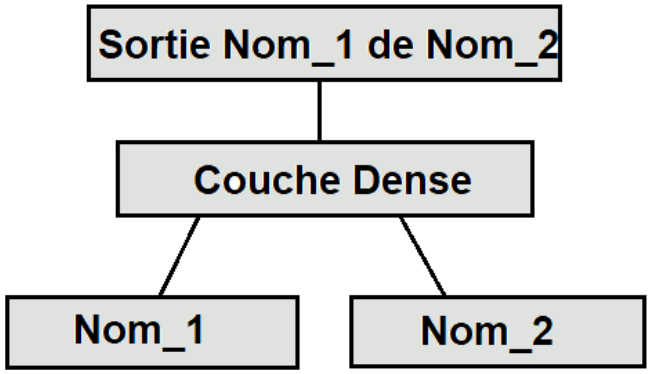

Figure 1: Multi-Layer Perceptron

et l'embedding attendu. L'entraînement de ce réseau se fait sur 300 epochs mais on utilise aussi de l'EarlyStopping avec une patience à 3.

On construit deux jeux de données. Le premier ensemble (FixedN1) est construit en sélectionnant manuellement 13 noms. Cet ensemble permet d'observer comment performe notre modèle sur des exemples précis. Mais aussi de construire facilement un ensemble de Test avec des N1 différents de l'ensemble de Train. Le second jeux de données est plus général (300N1300N2), avec 5000 occurrences différentes. Ces occurrences sont choisies pour être assez représentées dans le corpus initial et devrai avoir des embeddings de bonne qualité. De plus, cela permet d'observer les performances de notre modèle sur un ensemble plus varié.

L'ensemble FixedN1 est construit avec une liste de mot assez générique et peu idiomatique :

salle, outil, jeu, règle, équipe, chef, liste, acte, zone, technique, question, méthode, droit

On construit l'ensemble de test avec toutes les occurrences contenant "acte" et "technique". On rajoute à cela 10% des occurrences des autres noms. Cet ensemble de test sera divisé en deux parties pour obtenir un ensemble de développement et un ensemble d'évaluation. Le reste de nos occurrences partent dans notre ensemble de train.

Pour construire l'ensemble 300N1300N2 on commence par conserver les 300 noms les plus présents en position N1, on appelle cet ensemble 300N1. On regarde ensuite les 300 noms les plus présents en position N2 et qui ont en position N1 un élément de l'ensemble 300N1. Notre ensemble 300N1300N2 est

l'ensemble des occurrences ayant un élément de 300N1 en position N1 et un élément de 300N2 en position N2.
On répartit l'ensemble 300N1300N2 en 3 parties Train/Dev/test avec (70/15/15)% des données totales.

2.2 Méthode d'évaluation du modèle

Pour évaluer la qualité de notre MLP on va procéder de la façon suivante :
Pour chaque couple (N1,N2) notre MLP va produire une forme construite "N1-de-N2". On évalue la proximité de l'embedding produit par rapport à l'embedding de "N1 de N2".
Pour cela, on compare "N1-de-N2" à toutes les formes construites de notre jeu de test. On trie nos formes construites du plus proche au moins proche de proximité à notre "N1-de-N2". Et enfin, on identifie le rang de la forme construite attendu.
Ce rang nous permet de calculer un MRR (Mean Reciprocal Rank)

$$\text{MRR}= \frac{1}{|Q|} \sum_{i=1}^{|Q|} \frac{1}{rang_i}$$

Ici Q est l'ensemble de Test.

Le MRR fournit des valeurs dans l'intervalle $]0, 1]$. De bons résultats donnent des valeurs proches de 1.

Le calcul du MRR est fait sur une partie avec des N1 déjà connus (N1Seen) et sur une seconde partie avec des N1 jamais vu à l'entraînement (N1NotSeen). De plus, pour éviter des biais dans la création de nos embeddings avec Word2Vec on double le corpus d'entrée avec une version sans formes construites. L'idée est de limiter la modification des embeddings des composants des formes construites. (Je ne sais pas où mettre ces deux phrases, en conclusion ? Plus haut ?)

2.3 Résultats et conclusions

Dans un premier temps on utilise FixedN1 pour l'entraînement et le test. Le jeu de test contient 434 couples $(N1, N2)$.

	FixedN1	300N1300N2	Train=300N1300N2 Test=FixedN1
Rang Moyen	31	2704	230
MRR N1NotSeen	0.0729	NA	NA
MRR N1Seen	0.5452	0.0019	0.0139

Les premières expériences réalisées correspondent à la première colonne. On constate que le modèle produit des résultats satisfaisants pour l'ensemble N1Seen, mais beaucoup moins pour N1NotSeen.

On conclut de cette première expérience que le modèle était capable de composer des embeddings, mais seulement pour ce qu'il a vu à l'entraînement.
Pour corriger cela on effectue la seconde expérience avec un ensemble d'entraînements et de tests beaucoup plus variés. Les résultats chutent drastiquement quand on lui demande cette généralisation. Cependant la difficulté de la tâche proposée n'est pas la même que dans l'expérience précédente. Il y a plus d'éléments dans l'ensemble d'évaluations et distinguer le bon devient plus difficile.
On corrige ce biais dans la troisième expérience, la difficulté de la tâche devient la même mais le modèle n'a pas réussi à apprendre une fonction pour composer des embeddings.
De plus, si l'on regarde en détail les embeddings produits par le modèle on constate qu'ils sont toujours proches des mêmes embeddings. Une des raisons qui pourrait expliquer cela est que le modèle construit des embeddings proches pour n'importe quelle entrée.
Une autre remarque concernant le MRR, aurait pu être intéressante sur des expériences avec de bons résultats. Cependant quand les résultats sont vraiment mauvais le MRR ne différencie pas assez le 200 ème rang du 400 ème rang.

Pour conclure, cette approche par concaténation ne produit pas les résultats voulus. Le MLP n'a pas réussi à apprendre une opération de composition d'embeddings.
Cette approche par concaténation nous oblige à utiliser un système externe (Word2Vec) pour générer nos embeddings de formes construites.
Dans la partie suivante, on va utiliser une autre approche qui permet d'avoir plus de contrôle sur les embeddings des formes intermédiaires.

3 L'approche compositionnelle

3.1 Principe général du modèle

L'approche compositionnelle consiste à produire l'embedding d'une forme construite dans la fonction de composition. Cette fonction est générée par un MLP qui prend la forme d'un auto-encodeur et possède une couche intermédiaire qui est l'embedding de notre forme construite.
L'embedding d'une forme construite n'est donc pas dans le même espace que les embeddings de ses composants, ses espaces peuvent avoir des dimensions différentes.
Cette caractéristique à l'avantage d'être itérative, on peut construire l'espace des formes Nom-Adjectif, puis l'utiliser pour produire l'espace des Déterminants-Nom-Adjectif. Bien que cette méthode est plus flexible, on rajoute la difficulté d'évaluer la "qualité" des embeddings intermédiaires.
On fixe deux propriétés que l'on aimerait observer dans les espaces produits :

Les mots ou les formes construites qui ont un sens proche, doivent être proches dans notre espace. (Stabilité)

Les formes construites doivent aussi garder une trace de leur composants, "voiture rouge" doit être une voiture qui est rouge. (Compositionnalité)

On travail dans cette partie sur des formes Nom-Adjectif ou Adjectif-Nom. On récupère les embeddings du nom et de l'adjectif avec l'aide de Word2Vec. On se référera à cet espace comme l'espace des mots simples. De plus, l'adjectif sera toujours considéré comme en seconde position. C'est à dire que "petite fille" sera fourni à notre modèle comme "fille petite".

Notre MLP prend la forme d'un auto-encodeur. On fournit les embeddings du nom et de l'adjectif en entrée. On les encodent dans une couche intermédiaire (N-ADJ) puis on coupe notre modèle en deux. La première partie reçoit en entrée la couche N-ADJ et l'embedding de l'adjectif pour obtenir en sortie l'embedding du nom. Et la seconde partie reçoit la couche N-ADJ et l'embedding du nom pour retrouver l'embedding de l'adjectif.

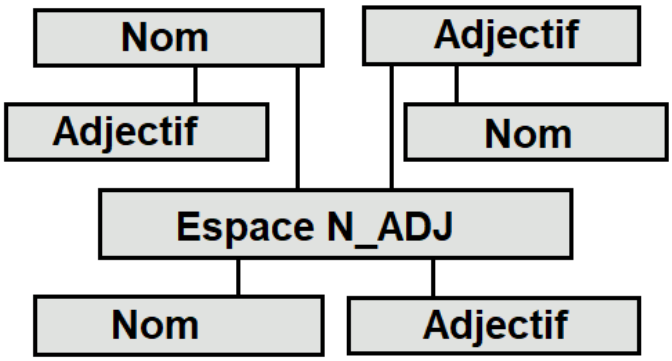

Figure 2: Auto-Encodeur

Les noms et les adjectifs sont encodés dans des embeddings de dimension 300. L'entrée de notre réseau est donc de dimension 600. La couche de l'espace N-ADJ est une couche Dense de dimension 300 avec une fonction "relu". Les couches de sorties sont aussi de dimension 300 et reçoivent en entrée la concaténation d'un embedding et de la couche N-ADJ, donc de dimension 600. La fonction de loss est une similarité cosinus entre le nom produit et le nom fournit en entrée. Le réseau est entraîné sur 50 epochs avec de l'EarlyStopping avec une patience à 3 et un delta de 0.001.

Pour les ensembles de données utilisées pour l'entraînement et l'évaluation on récupère toutes les occurrences de "Nom Adjectif" ou "Adjectif Nom". On les répartit en 3 parties FullTrain/FullDev/FullTest (70/15/15)% ou (162 000/35 000/35 000) données. On crée aussi 1000Test qui est composé des 1000 premières lignes de FullTest pour pouvoir évaluer certaines propriétés rapidement.

On va attendre plusieurs propriétés de notre modèle :

- Une cohérence dans l'encodage-décodage. Le modèle doit être capable de retrouver le nom et l'adjectif fournis en entrée.
- Un espace intermédiaire (N-ADJ) stable, avec des formes construites proches si elles ont un sens proche.

3.2 Métrique et résultats de l'encodage-décodage

Pour mesurer la capacité de notre modèle à encoder-décoder on utilise une mesure du type MRR. Pour chaque sortie de notre MLP on calcule les plus proches voisins dans l'espace des mots simples. Si le nom fourni en entrée est présent dans les plus proches voisins de notre sortie, alors on note son rang, idem pour l'adjectif. Avec ce rang on peut calculer un MRR, sinon on rajoute 0 à notre MRR.

Prenons un exemple où l'on prend les 4 plus proches voisins :

voiture	pomme	carotte
voiture	poire	cadeau
camion	pêche	gâteau
moto	orange	bougie
vélo	pomme	briquet

$MRR = (\frac{1}{1} + \frac{1}{4} + 0) \times \frac{1}{3} = \frac{5}{12}$

On applique cette méthode sur l'ensemble 1000Test :

- MRR des Noms : 1.0
- MRR des Adjectifs : 1.0

Les résultats obtenus montrent que notre système est toujours capable de retrouver les données reçues en entrée. Ce n'est pas si étonnant, on peut seulement dire que notre MLP est capable d'effectuer une petite compression, ce qui ne semble pas être une tâche très difficile. Notre auto-encodeur semble avoir la propriété d'être cohérent dans son encodage décodage. La prochaine étape est de regarder les propriétés de l'espace intermédiaire produit.

3.3 Évaluation de la stabilité de l'espace N-ADJ

3.3.1 Approche naïve

Évaluer l'espace produit par notre auto-encodeur suppose que l'on ait une idée précise des propriétés attendues. On a fini par sélectionner l'idée que les mots avec un sens proche doivent être proches, ce que l'on appelle la stabilité. On va réduire cette propriété à "les formes construites qui ont un composant en commun doivent être proches".
On aimerait donc que "pomme jaune", "pomme verte" et "pomme rouge" soient proches, et "voiture blanche" soit éloigné de nos trois pommes.

On évaluera la proximité de deux embeddings avec une similarité cosinus.

L'une des méthodes les plus simples est de regarder les plus proches voisins de points au hasard. Cette méthode ne produit pas de mesure générale sur notre espace. De plus cette méthode dépend grandement du nombre de points présents dans notre espace, un gros jeu de tests rend la tache plus difficile.

On commence par effectuer les 5 voisins les plus proches, avec 1000Test

rang	fonction-public	directeur-général	collectivité-local	niveau-haut
0	fonction-public	directeur-général	collectivité-local	niveau-haut
1	action-public	administration-général	entreprise-local	fréquence-haut
2	sphère-public	conseiller-général	fiscalité-local	performances-haut
3	utilité-public	cadre-général	acteur-local	fonctionnaire-haut
4	chaîne-public	intérêt-général	association-local	plateau-haut

Sur ce jeu de tests on observe la propriété de stabilité. Les formes composées les plus proches ont toutes un composant en commun. Le composant commun est l'adjectif, il est possible d'obtenir le nom comme composant commun en modifiant la loss. Il faut augmenter le poids de la similarité cosinus des noms par rapport à celle des adjectifs.
On répète cette expérience mais avec l'ensemble FullTest. L'ensemble contient beaucoup plus de points donc c'est un test plus difficile.

rang	fonction-public	directeur-général	collectivité-local	niveau-haut
0	fonction-public	directeur-général	collectivité-local	niveau-haut
1	poste-personnel	tranche-ferme	entreprise-local	du-cru
2	recherche-génétique	information-contradictoire	institution-local	formation-compétent
3	abus-premier	prévision-saisonnier	fiscalité-local	gauche-caviar
4	stabilisation-électronique	contrôleur-général	école-alsacien	generated-content

Les résultats obtenus sont très différents du test précédent. On voit que notre espace ne vérifie plus la propriété de stabilité. Ces expériences nous montrent que notre modèle est effectivement capable de rapprocher les formes construites avec un composant commun. Néanmoins, le modèle n'est pas capable de le faire de manière suffisamment précise pour un jeu de données plus grand.
Il est assez surprenant que l'on ai notre propriété à ce stade. L'auto-encodeur utilisé n'a aucune indication que cette propriété est désirable. On obtient cette propriété car deux formes composées qui ont un composant commun, ont une représentation proche dans l'entrée de notre MLP. On va donc modifier notre auto-encodeur pour essayer d'obtenir la propriété de stabilité sur des gros jeux de données.

3.3.2 Auto-Encodeur amélioré et comparaison d'espace

On aimerait que notre modèle prenne en compte la proximité de deux mots dans l'espace initial et essaye de la transposer dans l'espace intermédiaire des N-ADJ. Pour cela on va doubler notre réseau de neurones pour avoir deux couples $(N1, A1)$ et $(N2, A2)$ en entrée et en sortie.

(Schéma à faire)

On modifie notre loss pour que si $N1$ et $N2$ ou $A1$ et $A2$ sont proches, alors $N1A1$ et $N2A2$ doivent être proches. On a donc choisi cette expression :

$loss_R = 0.4 \times \cos(N_1, N_1') + 0.1 \times \cos(A_1, A_1') + 0.4 \times \cos(N_2, N_2') + 0.1 \times \cos(A_2, A_2')$
$Loss = loss_R - (1 - |\cos(N1A1, N2A2) - (0.8 \times \cos(N1, N2) + 0.2 \times \cos(A1, A2)|)$

La fonction cosinus représente une similarité cosinus. $loss_R$ correspond à la partie de loss qui était présente dans l'auto-encodeur précédent.
Les coefficients 0.8,0.4 et 0.1 servent à favoriser la proximité des noms. On peut regarder comment cette modification se comporte sur des valeurs fixes pour vérifier que cela fait effectivement ce que l'on attend :

$A = \cos(N1A1, N2A2)$	$B = \cos(N1, N2) + \cos(A1, A2)$	$1 - \lvert A - B \rvert$
0	0	1
0	1	0
1	0	0
1	1	1

Cette table de vérité montre que si nos noms ou nos adjectifs sont proches/éloignés alors nos formes composées doivent être proches/éloignées pour minimiser la loss. Cette fonction devrait encourager notre espace intermédiaire à être stable.

On commence par effectuer les 5 voisins les plus proches, avec 1000Test :

rang	fonction-public	directeur-général	collectivité-local	niveau-haut
0	fonction-public	directeur-général	collectivité-local	niveau-haut
1	fonction-commercial	directeur-commercial	entreprise-local	niveau-élevé
2	fonction-autre	directeur-schéma	fiscalité-local	niveau-même
3	action-public	administration-général	association-local	niveau-technique
4	sphère-public	direction-même	acteur-local	niveau-national

On observe que sur un petit jeu de tests l'espace intermédiaire est stable. On note aussi que les noms sont favorisés par rapport aux adjectifs, ce qui était effectivement encouragé dans la loss.
Passons au second ensemble de test FullTest avec 35 000 points.

rang	fonction-public	directeur-général	collectivité-local	niveau-haut
0	fonction-public	directeur-général	collectivité-local	niveau-haut
1	flûte-enchanté	devis-multiple	collection-archéologique	niveau-élevé
2	financement-conséquent	directeur-juridique	client-newsletter	niveau-quatrième
3	fluide-interne	directeur-commercial	collectif-départemental	nature-prospectif
4	fonction-commercial	devi-technique	carte-officiel	n-alert

Les résultats obtenus semblent meilleurs que les précédents, mais il est difficile de dire de combien. Il faudrait développer une métrique capable de mesurer la stabilité de notre espace, et de comparer cette stabilité à d'autres espaces d'embeddings.

On utilise donc une mesure de clusters qui fait l'hypothèse que : Si la propriété de stabilité est vérifiée, alors la majorité des formes construites avec un composant commun doivent être proches. On peut alors calculer le centre de masse de ces formes construites, et regarder la proximité par rapport à ce centre de masse. La proximité la moins élevée correspond à la taille du cluster.

Prenons un exemple :

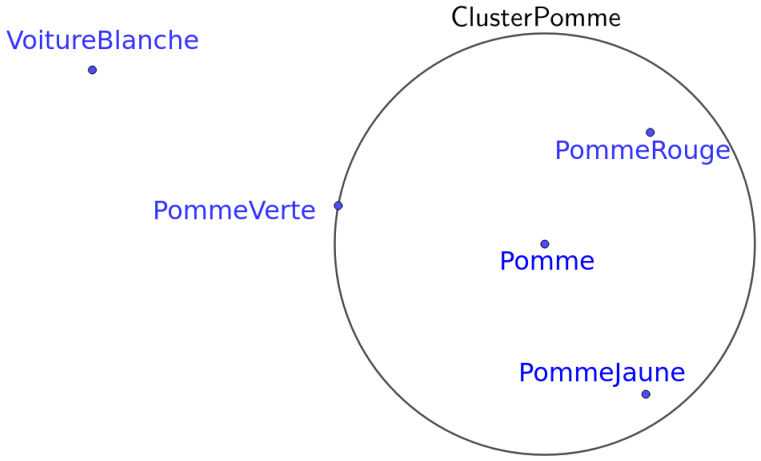

Figure 3: Illustration de la méthode des clusters

Le rayon du cercle "ClusterPomme" correspond ici à la taille de ce cluster. Si l'on systématise cette mesure pour tous les noms et adjectifs on obtient la taille moyenne de nos clusters. Une remarque que l'on peut avoir est que le point "Pomme" n'existe pas dans notre espace c'est uniquement le centre de masse de nos occurrences.

On utilise cette métrique pour comparer les espaces intermédiaires produits par nos deux auto-encodeurs

	auto-encodeurV1	auto-encodeurV2
Similarité Noms	0.9035	0.9050
Similarité Adj	0.8947	0.8949
Distance Noms	2652.75	2958.93
Distance Adj	2395.90	2671.02

Les résultats sont très similaires, on observe uniquement une différence au niveau de la taille en norme 1 de nos clusters qui sont apparemment plus grand

dans auto-encodeurV2.
On n'a pas vraiment d'explication à cette différence, il faudrait sûrement avoir plus d'informations sur l'espace intermédiaire pour arriver à faire des comparaisons entre des espaces. Je ne pense pas que ces mesures permettent d'invalider le second auto-encodeur mais plutôt que nos métriques ne sont pas les bonnes.

Pour conclure, cette approche a apporté d'importants bénéfices en comparaison à la première. Pouvoir ajuster l'espace des formes construites selon plusieurs critères permet une grande flexibilité. Au niveau des résultats il n'est pas possible de dire que l'on a réussi à construire de manière satisfaisante des embeddings de Nom-Adjectif.
Il faudrait développer des méthodes pour évaluer les embeddings des formes intermédiaires.

4 Pistes d'idées non aboutis/implémenté

4.1 Études des espaces intermédiaires

L'approche compositionnelle crée des espaces intermédiaires de formes construites. On évalue ces espaces avec une propriété de stabilité : "Deux formes construites proches doivent avoir un sens proche". Cette formulation manque de précision sur ce qu'est "un sens proche". Expliciter ce que l'on attend de nos espaces intermédiaires aiderait à produire de meilleures métriques. Pour l'instant la métrique des mesures de clusters permet de comparer les performances de deux espaces intermédiaires.

De plus cette mesure de clusters présente plusieurs problèmes :

- On ne considère pas les points à l'intérieur du cluster. On pénalise de la même façon la proximité entre "fonction-publique" et "poste-personnel" par rapport à "directeur-général" et "prévision-saisonnier"
- La mesure des clusters actuels se base sur une mesure de distance (euclidienne ou similarité cosinus). Cela complique la comparaison avec d'autres espaces. En effet, notre modèle nous donne très peu de contrôle sur les caractéristiques de l'espace produit. On pourrait imaginer une approche KNN où N est le nombre d'éléments dans l'ensemble de test ? Cela pourrait permettre de s'affranchir de certaines spécificités de l'espace intermédiaire.

On pourrait aussi essayer de modifier la loss de notre modèle pour améliorer les résultats de notre mesures de clusters. En fournissant régulièrement en entrée de notre modèle deux formes construites avec un élément en commun ("fonction-public" et "fonction-commercial"). Cependant, je pense qu'il faudrait d'abord élaborer des métriques de qualité pour ensuite essayer d'améliorer nos résultats sur une métrique donnée.

Enfin, notre modèle fait l'hypothèse que les entrées fournies sont compositionnelles. Il faudrait peut être faire un pré-traitement de nos données pour enlever les formes non compositionnelles. On peut s'appuyer sur ce travail. `https://aclanthology.org/J19-1001/`

4.2 Systématisation à la syntaxe X-Barre

Ce modèle a été imaginé pour être itéré en respectant un arbre de syntaxe. On aimerait donc produire des espaces intermédiaires de N-ADJ mais aussi de DET-N-ADJ ce qui construit un espace des groupes nominaux simples. Cette approche nous confronte à plusieurs problèmes :

- Le modèle actuel ne semble par marcher si l'on considère que l'on met des ϵ dans les feuilles de notre arbre. Il serait pratique d'avoir un embedding du mot vide pour construire une représentation de N-ϵ. En l'absence de cette représentation on a supposé que N-ϵ était le barycentre de nos N-ADJ avec N fixé.
- Notre méthode de construction des espaces intermédiaires implique qu'un groupe nominal de la forme DET-N-ADJ ou DET-N-ADJ-ADJ ne sont pas dans le même espace. On obtient un espace différent pour chaque syntaxe de groupe nominal. On pourrait essayer "d'aligner" nos différents embeddings de groupes nominaux ?

Une systématisation à la syntaxe X-Barre me semble difficile tant que l'on n'a pas plus de connaissances sur nos espaces intermédiaires.

Le code utilisé pour réaliser ce travail est disponible sur GitHub : `https://github.com/Ounaye/TAL_ApprentissageComposition_NADJ`