analyse_simul_gensim

March 22, 2020

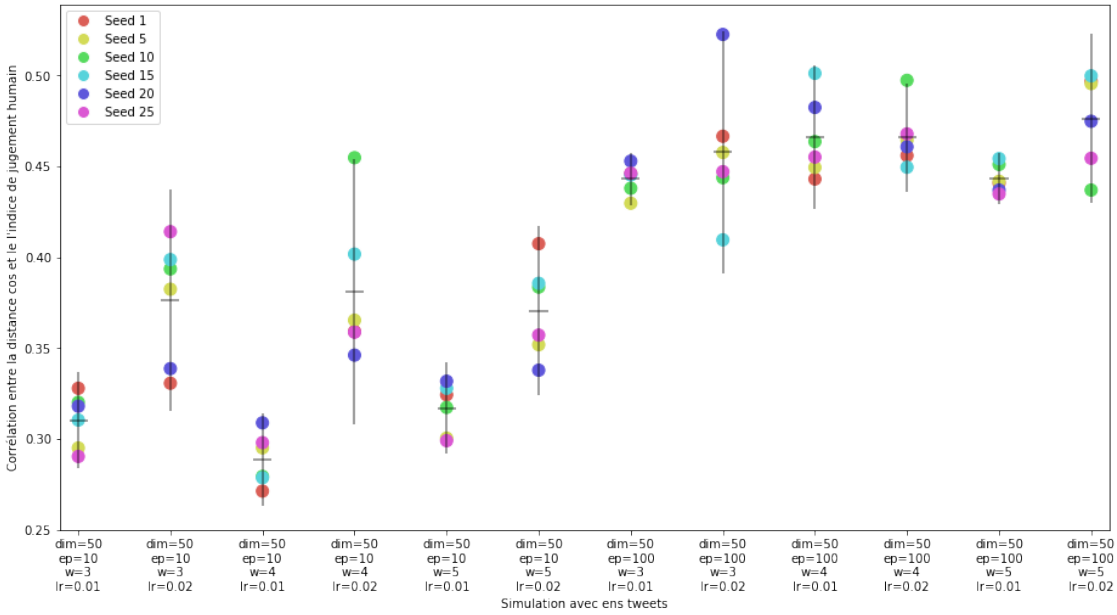
1 Evaluation des paramètres à choisir pour le modèle word2vec de Google

1.1 Nombre d'epochs, learning rate et window

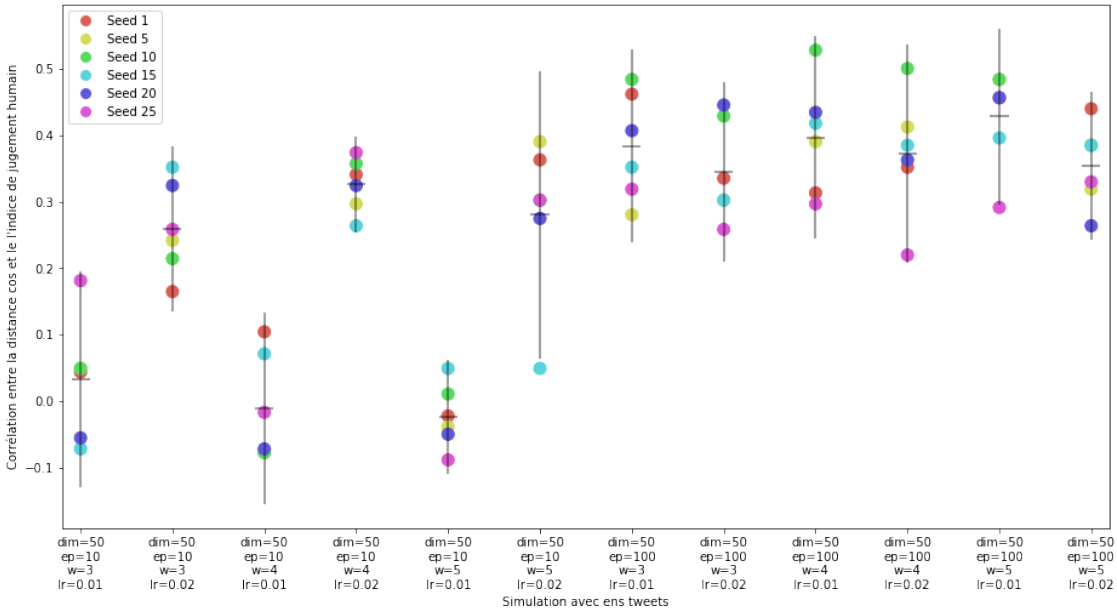
On lance sur l'ensemble des tweets avec une dimension des word-embedding de 50 et on mesure la corrélation entre le human judgement et la similarité cosinus.

On teste ici :

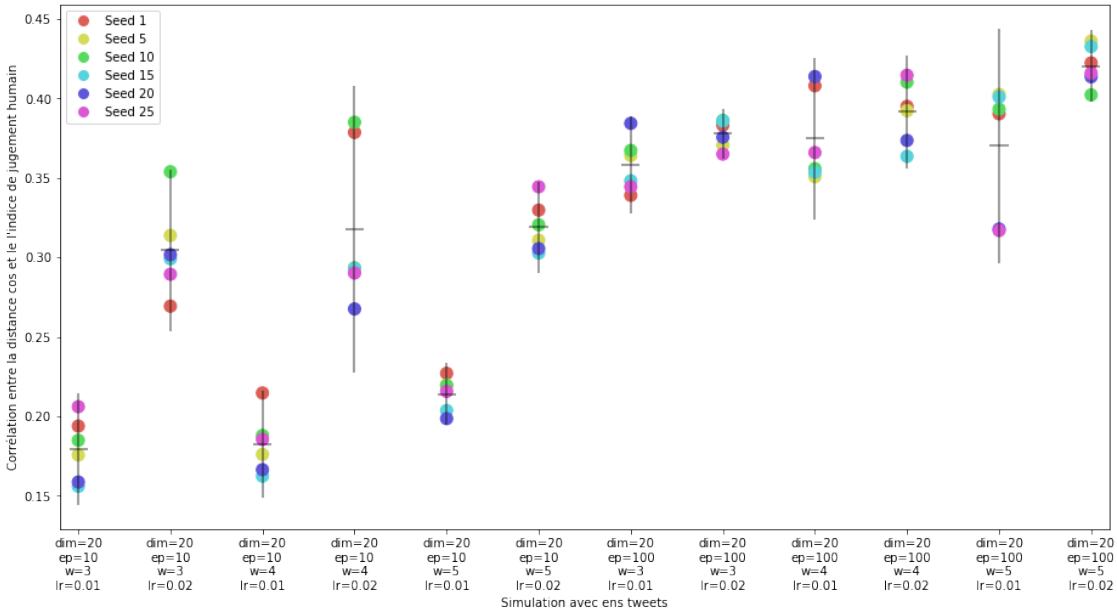
- **le nombre d'époch**s : Il y a un net effet du nombre d'époch : passer de 10 à 100 epochs fait augmenter le score de corrélation.
- **le learning-rate** : Le learning-rate 0.02 semble donner systématiquement de meilleurs résultats que 0.01
- **la window** : La taille de la window ne semble pas jouer un rôle majeur, ce paramètre dépend beaucoup des autres paramètres choisis. Pour 100 epoch et un learning rate de 0.02, c'est la taille de fenêtre 5 qui semble en moyenne donner le meilleur résultat mais ce résultat n'est pas significativement meilleur que les autres.



En faisant tourner une nouvelle fois le modèle sur seulement 100 000 tweets pour voir si les résultats de choix du modèles sont aussi clairs avec moins de tweets, les effets observés sur l'ensemble des tweets se confirment.



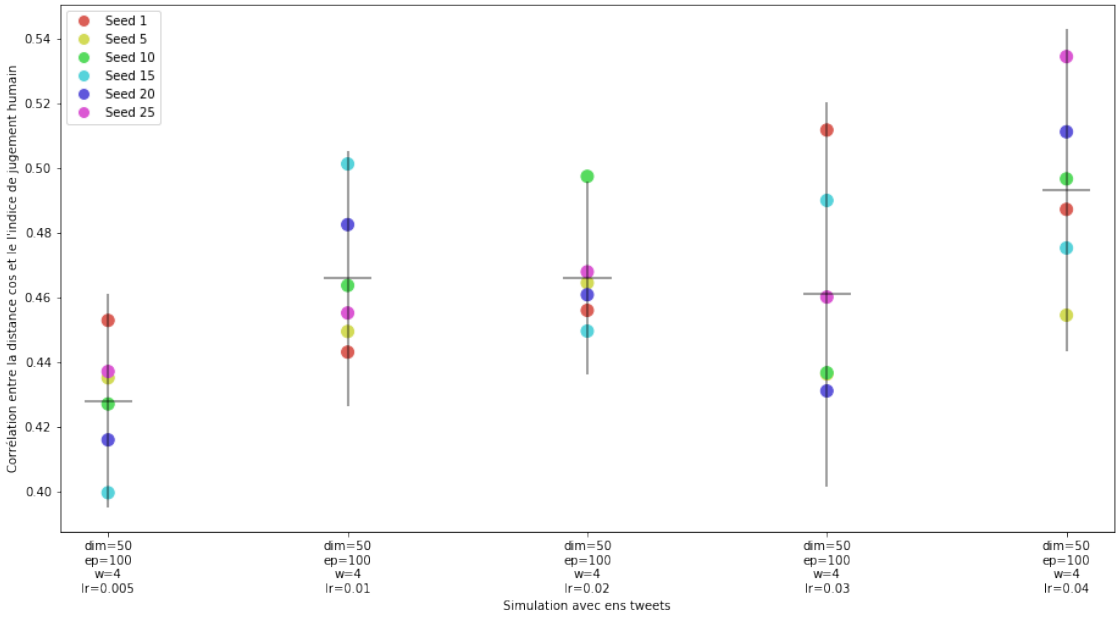
En faisant tourner une nouvelle fois le modèle sur l'ensemble des tweets mais avec des word-embedding de dimension 20, les effets observés sur l'ensemble des tweets se confirment encore une nouvelle fois !



1.1.1 Learning rate

Nous venons de voir que le learning rate de 0.02 donnait de meilleur résultats que 0.01 dans les comparaisons que nous avons effecutées précédemment. Voyons désormais à quel point il joue lorsque nous le faisons varier davantage en fixant par exemple le modèle sur l'ensemble des tweets tel que dim = 50, ep = 100 et w = 4.

On remarque que même si le paramètre lr = 0.04 donne, ici, de résultats un peu meilleurs que les autres, la moyenne de la corrélation est supérieure d'environ seulement 0,03 par rapport à lr = 0.02 et le gain n'est pas significatif. Nous pouvons donc maintenir notre choix de lr = 0.02 sans perdre en efficacité.



1.1.2 Window

Nous venons de voir que, selon les autres paramètres, la taille de la window optimale varie. Il semble alors difficile de se prononcer sur le "meilleur choix" de window à effectuer.

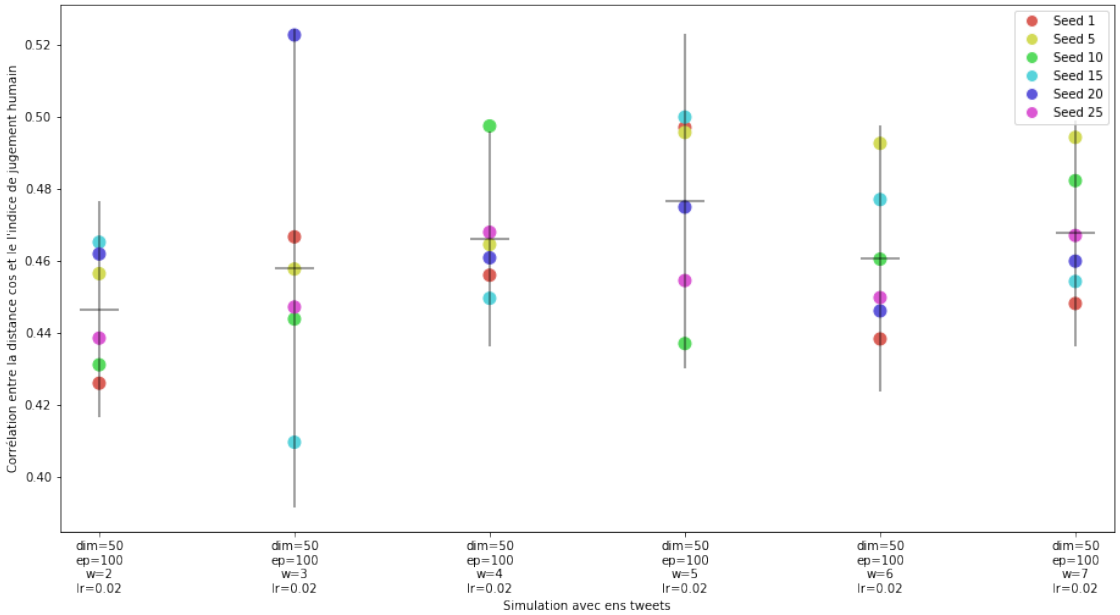
En effet, certains travaux ("Dependency-Based Word Embeddings", Levy & Golberg, 2014) indiquent que, suivant la taille de fenêtres que nous choisissons, les informations capturées sont différentes :

- Les **grandes** fenêtres ont tendance à capturer plus d'informations sur le domaine du mot : les autres mots (de tout type) qui sont utilisés dans les discussions connexes.
- Les **petites** fenêtres ont tendance à mieux saisir le mot lui-même : les autres mots qui sont fonctionnellement similaires : ses extensions (avec préfixes / suffixes différents), ses synonymes ou les alternatives qui pourraient se substituer au mot d'origine.

Cela pourrait alors expliquer la difficulté de choisir "la bonne" taille de fenêtre.

Nous comparons ici davantage de tailles de fenêtre, en fixant par exemple le modèle sur l'ensemble des tweets tel que dim = 50, ep = 100 et lr = 0.02.

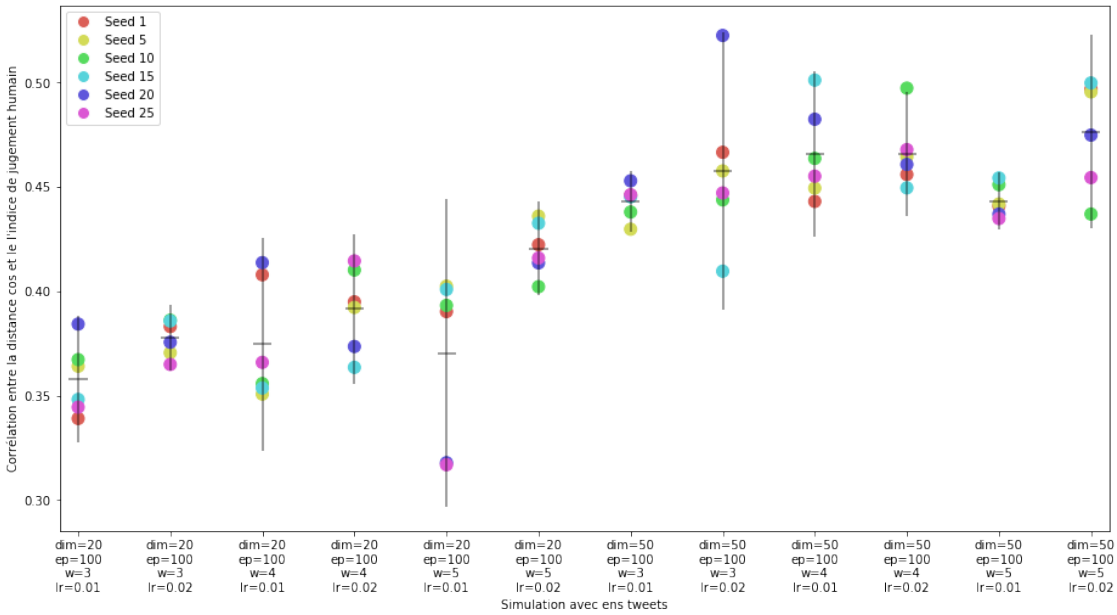
On remarque que même si le paramètre w = 5 donne, ici, de résultats un peu meilleurs que les autres, la moyenne de la corrélation est supérieure d'environ seulement 0,01 par rapport à w=4 et le gain n'est pas signficatif. Nous pouvons donc maintenir notre choix de w=4 sans perdre en efficacité.



1.2 Dimension des vecteurs-mots

1.2.1 Une dimension de taille 50 qui donne de meilleurs résultats que celle de taille 20...

On cherche cette fois-ci à évaluer l'effet de la dimension des web-embedding. Augmenter la dimension permettrait de projeter avec plus de précision les mots dans des espaces de dimension plus grand et donc de mieux capter les relations entre les mots. En revanche, une dimension trop élevée pourrait également dégrader la qualité de la projection vectorielle en rajoutant plus de bruit. En pratique, pour toutes les combinaisons de paramètres, on observe une amélioration des résultats obtenus en dimension 50 par rapport en dimension 20.



1.2.2 ... avec toutefois de moins bons résultats lorsque l'on augmente la dimension au delà de dim = 300

Nous avons ensuite analysé les résultats pour des dimensions supérieures à 100, afin de voir si, à partir d'une certaine taille de word-embedding, les résultats finissent par se dégrader.

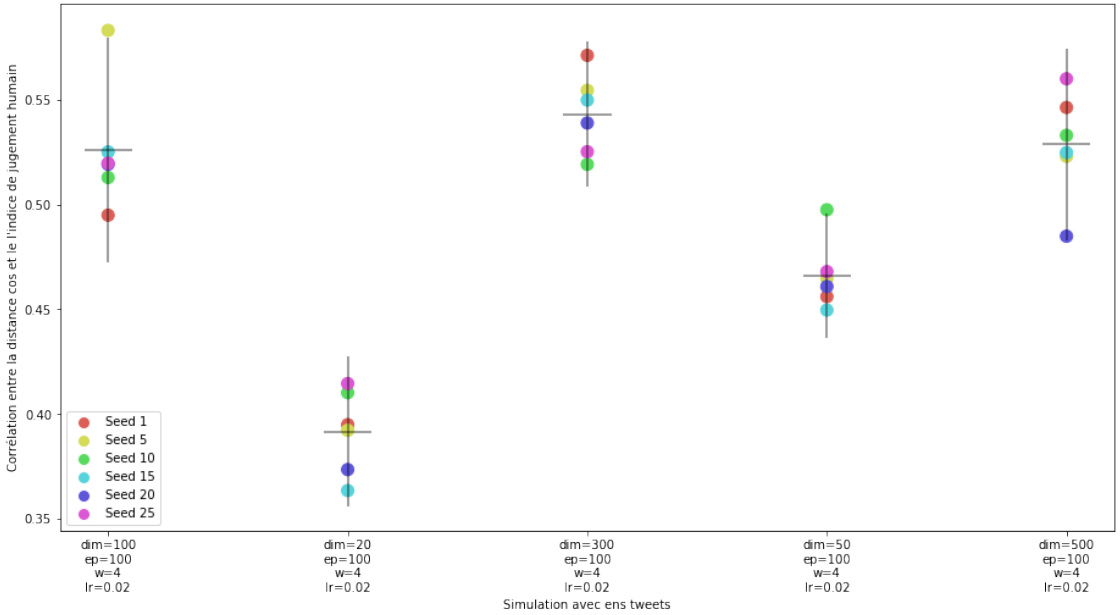
En effet, selon des recherches ("Glove: Global Vectors for Word Representation" Pennington, Socher & Manning), la qualité des représentations vectorielles s'améliore à mesure que l'on augmente la taille du vecteur, mais seulement jusqu'à atteindre 300 dimensions. Après 300 dimensions, la qualité des vecteurs commence à diminuer.

Bien sûr, ce seuil de 300 peut dépendre des données utilisées. La dimension du vecteur doit également être adaptée à la taille du vocabulaire. Par exemple, avec un vocabulaire d'une centaine de mots, il serait inefficace d'utiliser des projections en grande dimension. Il y a en effet un risque d' "overfitting" (erreur d'entraînement nulle) qui rend l'erreur de test trop élevée (quand on s'intéresse à un autre corpus que le notre, cf. cours d'introduction au machine learning). Un des articles fondateurs de word2vec ("Efficient Estimation of Word Representations in Vector Space", Mikolov & Chen, 2013) recommande donc d'augmenter à la fois la dimensionnalité des vecteurs et la quantité de données d'apprentissage. Ils nous avertissent sur le point suivant : "Bien que cette observation puisse sembler banale, il convient de noter qu'il est actuellement populaire de former des vecteurs de mots sur des quantités de données relativement importantes, mais avec une taille insuffisante."

Le choix de la dimension dépend également de la puissance de calcul de l'ordinateur : même si une dimension de 300 donne des résultats plus fiables, il peut être intéressant de réduire la dimension si la machine utilisée est trop lente à calculer les vecteurs.

Nous comparons ici davantage l'effet de dimension, en fixant par exemple le modèle sur l'ensemble des tweets tel que ep = 100, w=4 et lr = 0.02. Comme dans l'article cité ci-dessus, On observe

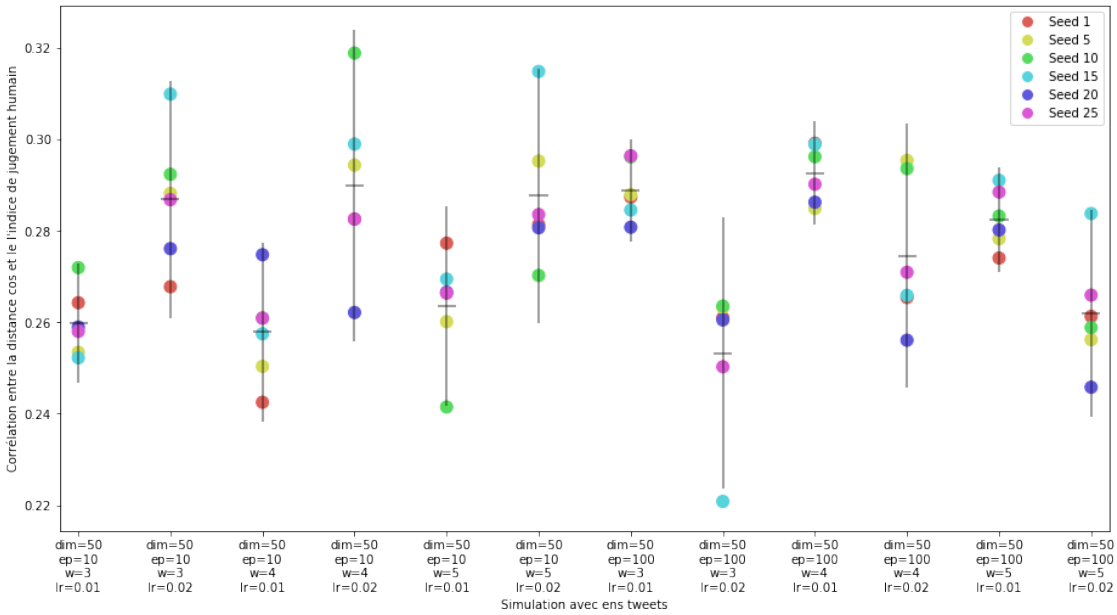
une augmentation de l'efficacité du modèle jusqu'en dimension 300 et une efficacité moindre en dimension 500. Bien que l'efficacité du modèle semble meilleure en dimension 300, nous choisissons la dimension 100, plus rapide à tourner et d'un gain qui n'est pas significativement inférieur.



1.3 Evaluation avec la distance euclidienne et non la dissimilarité cosinus ?

On cherche cette fois-ci à voir l'effet de prendre (l'inverse d') une distance euclidienne (sur vecteurs normalisés) plutôt qu'une similarité cosinus. On remarque que cette fois-ci les résultats sont instables.

- **le nombre d'époch**s : Il ne semble avoir aucun effet sur les résultats obtenus (étrange !)
- **le learning rate** : Le learning rate 0.02 donne bien des meilleurs résultats que 0.01 sur 10 epochs, mais sur 100, c'est l'inverse ! (étrange !)
- **la window** : Ce paramètre, comme avec la distance cosinus, n'a pas d'effet net mais le paramètre window = 4 semble donner des résultats légèrement meilleurs pour le learning rate = 0.02.



1.4 Conclusions

Ces différentes simulations nous amènent à choisir les paramètres suivants :

- **dim = 100** (résultat proche de dim = 300 mais plus rapide à tourner)
- **ep = le plus possible** (a priori une ou plusieurs centaines)
- **w = 4** (taille moyenne pour capter tous les types de proximité)
- **lr = 0.02** (meilleur que 0.01 dans l'ensemble des premières simulations et proche des autres lr dans le test comparant à davantage de lr)