Complémentarités de représentations vectorielles pour la similarité sémantique

Julien Hay[*,**], Tim Van de Cruys[***]
Philippe Muller[***] Bich-liên Doan[**,****]
Fabrice Popineau[**,****] Lyes Benamsili[*]

[*]Octopeek
22 Rue du Général de Gaulle, 95880 Enghien-les-Bains, France
https://www.octopeek.com
[**]LRI
Bat 650 / 660, Rue Noetzlin, 91190 Gif-sur-Yvette, France
[***]IRIT
Université Toulouse III Paul Sabatier
118 Route de Narbonne, 31062 Toulouse, France
[****]CentraleSupélec
3 rue Joliot-Curie, 91192 Gif-sur-Yvette, France

Résumé. La tâche de similarité sémantique textuelle consiste à exprimer automatiquement un nombre reflétant la similarité sémantique de deux fragments de texte. Chaque année depuis 2012, les campagnes de *SemEval* déroulent cette tâche de similarité sémantique textuelle. Cet article présente une méthode associant différentes représentations vectorielles de phrases dans l'objectif d'améliorer les résultats obtenus en similarité sémantique. Notre hypothèse est que différentes représentations permettraient de représenter différents aspects sémantiques, et par extension, d'améliorer les similarités calculées, la principale difficulté étant de sélectionner les représentations les plus complémentaires pour cette tâche. Notre système se base sur le système vainqueur de la campagne de 2015 ainsi que sur notre méthode de sélection par complémentarité. Les résultats obtenus viennent confirmer l'intérêt de cette méthode lorsqu'ils sont comparés aux résultats de la campagne de 2016.

1 Introduction

De nombreux travaux récents s'intéressent à la similarité sémantique, soit entre mots, soit entre groupes de mots, depuis les syntagmes jusqu'à des documents complets, en passant par la similarité entre phrases. Rapprocher des mots ou des phrases par leur sens permet d'utiliser des traits sémantiques dans des modèles en évitant la dispersion inhérente liée à la taille du vocabulaire, ou à l'espace des phrases possibles. La plupart des travaux dans ce sens calculent des similarités entre des représentations construites sur des bases distributionnelles, c'est à dire où la similarité de sens dérive d'une similarité des contextes d'apparition des mots, une hypothèse énoncée par Harris (1954).

Les représentations prennent la forme de vecteurs, matrices et tenseurs distributionnels (Turney et Pantel, 2010), où les dimensions correspondent à des co-occurrences lexicales (Curran, 2004), syntaxiques (Baroni et Lenci, 2010), ou bien des transformations de ces contextes, par réduction de dimension (Pennington et al., 2014) ou par l'intermédiaire d'apprentissage avec des réseaux de neurones (Mikolov et al., 2013). L'évaluation de ces modèles vectoriels repose soit sur des tâches externes où ils sont mis en jeu, soit sur des mesures intrinsèques, fondées sur des échantillons de mots similaires, ou des groupes de mots similaires.

Plus récemment, la notion de similarité sémantique textuelle motive des représentations vectorielles au delà de mots seuls. La représentation phrastique peut se construire par composition des représentations lexicales (Mitchell et Lapata, 2008; Van de Cruys et al., 2013) ou là encore être construite par l'intermédiaire d'un apprentissage par réseau de neurones (Le et Mikolov, 2014).

Les méthodes appliquées à la reconnaissance de paraphrases ou de similarités s'appuient donc sur des représentations vectorielles et sur différentes façons de les combiner, et utilisent aussi des appariements des éléments de phrase (Sultan et al., 2015). Un problème crucial pour ces représentations vectorielles est l'existence de nombreux hyper-paramètres dans leur construction et dans leur combinaison pour la définition de la similarité textuelle. De même, il existe peu de travaux qui tentent de combiner différentes représentations pour tirer parti d'éventuelles complémentarités des choix effectués en amont.

Nous présentons ici un travail sur la combinaison des représentations vectorielles pour explorer le potentiel de l'association de ces représentations à différentes échelles. Nous revenons dans la section suivante plus en détail sur la tâche de similarité sémantique textuelle avec un bref état de l'art de la campagne *SemEval*. La section 3 présentera nos motivations et hypothèses qui justifieront cette recherche de complémentarité des représentations vectorielles. Nous détaillerons ensuite notre méthode de recherche de complémentarité à travers deux algorithmes optimisant deux critères différents en section 4. Enfin, nous discuterons des résultats obtenus en les comparant aux résultats de la campagne de l'année 2016.

2 Similarité sémantique textuelle

Nous revenons brièvement sur les modèles vectoriels de mots et de phrases, pour discuter de leur place dans la tâche spécifique de mesure de similarité textuelle.

2.1 Modèles vectoriels

La représentation du texte dans un espace vectoriel est une technique de plus en plus utilisée ces dernières années dans plusieurs disciplines du *TAL*, comme en analyse de sentiment et en traduction automatique (Le et Mikolov, 2014). Cette représentation permet de « rapprocher » des mots, des phrases, et de manière générale, du texte, sans passer par une représentation précise de tous les éléments du sens. La représentation vectorielle permet ainsi de construire une forme pouvant positionner les mots les uns par rapport aux autres.

Chaque mot étant réduit à un vecteur de nombres, il est possible de calculer une similarité avec des mesures simples telle que la similarité cosinus. La construction des représentations vectorielles est possible par apprentissage non-supervisé sur de larges *corpus*. Les différentes

techniques de construction de ces vecteurs tiennent compte, pour chaque mot, de leurs voisins dans le texte, c'est à dire des mots qui sont proches dans une même phrase : le « contexte ».

Les vecteurs se prêtent également bien à différentes méthodes de composition. Il est possible de composer plusieurs mots afin de construire le vecteur d'une phrase en faisant la moyenne des éléments des vecteurs de tous les mots. D'autres méthodes s'attaquent directement à la représentation de phrases sans passer par la composition de vecteurs de mots (Le et Mikolov, 2014). Des approches supervisées permettent également la représentation de phrases grâce notamment aux réseaux de neurones profonds tels que les *recursive networks*, les *recurrent networks*, ou encore les *convolutional networks*.

2.2 La tâche *SemEval STS*

La tâche *STS* propose de mesurer quantitativement la similarité sémantique de deux phrases sur une échelle de 0 à 5, la valeur 5 signifiant que les phrases sont strictement identiques sur le plan sémantique. Chaque paire de phrases mise à disposition depuis 2012 a été évaluée par plusieurs annotateurs humains rémunérés via la plate-forme de *microworking Amazon Mechanical Turk*, en éliminant les annotateurs peu fiables et les phrases dont la mesure montre trop de variance. À chaque paire de phrases correspond une note moyenne, qui peut donner des valeurs réelles dans tout l'intervalle *[0, 5]*.

Ce sont au total plus de $14\ 000$ paires de phrases qui ont été mises à disposition dans le cadre de cette tâche depuis 2012. Le score d'un système est calculé, par la corrélation de Pearson entre leurs résultats et les données de test annotées pour l'année courante. Chaque équipe peut proposer trois systèmes et les entraîner sur les données des années antérieures.

De manière générale, les participants à la tâche l'abordent comme un problème de régression supervisée, et utilisent des descripteurs classiques comme le nombre de mots en commun, de séquences de différentes longueurs en commun, etc. Des mesures de similarité basées sur un alignement de mots et des mesures provenant du domaine de la traduction automatique sont aussi utilisées. En 2016, les trois participants vainqueurs ont exploité de nouvelles techniques de représentations vectorielles de phrases, toutes basées sur des architectures de réseaux de neurones profonds comme Rychalska et al. (2016) qui ont utilisé un *Recursive Neural Network*.

3 Motivations

3.1 Hypothèse

Une comparaison basique entre deux phrases consisterait à calculer leur similarité topicale et lexicale. D'autres méthodes provenant de la sémantique distributionnelle permettent, quant à elles, de représenter l'ensemble de la phrase dans un espace sémantique commun à toutes les phrases d'un corpus donné. Ces méthodes demandent le prétraitement d'un corpus et de paramétrer l'algorithme qui apprend à représenter chaque phrase. Dans cet article, nous proposons une méthode qui tente de représenter plus finement les phrases sans se restreindre à un paramétrage précis. En l'occurrence, nous pensons que l'humain est capable de cibler différents aspects sémantiques dans l'objectif de comparer des morceaux de texte sur plusieurs plans.

Ces différents aspects peuvent, par exemple, être le sujet traité (*topic*), l'action dans la phrase, le mouvement, les entités impliquées, les informations spatio-temporelles, etc.

En *TAL*, on cherche en général à trouver le meilleur algorithme qui génère les représentations vectorielles, ou on essaye d'optimiser les paramètres de certains algorithmes. À notre connaissance, aucun travail ne tente de détecter automatiquement les aspects sémantiques qui permettraient une comparaison textuelle optimale de niveau humain.

C'est ce que nous proposons de faire grâce à la variation d'un certain nombre de paramètres de prétraitement de phrases et de construction de leur vecteur représentatif. Nous évaluons notre méthode sur la tâche de *Semantic Textual Similarity* (*STS*) des campagnes *SemEval*. Et afin de capturer automatiquement des aspects sémantiques permettant une comparaison optimale, nous optimisons la sélection de différentes représentations vectorielles sur le critère de la complémentarité dans la tâche de *STS*. Cette variation peut permettre de cibler différents aspects de la phrase, et ainsi faire un jugement de pertinence au plus proche de l'humain.

Dans notre article, nous définissons une suite de représentations vectorielles les plus complémentaires possibles comme une suite de représentations, qui, associées, permettent d'obtenir les meilleurs résultats sur la tâche en exploitant une diversité dans l'affectation des paramètres de prétraitement du corpus et de construction des vecteurs.

Nous proposons d'utiliser une extension de *Word2Vec* (Mikolov et al., 2013), communément appelée *Doc2Vec* (Le et Mikolov, 2014) permettant de représenter un document ou un ensemble de phrases dans un espace sémantique. Nous avons utilisé l'implémentation *Doc2Vec* de *Gensim* (Řehůřek et Sojka, 2010).

3.2 Variation de paramètres

Notre contribution se fonde sur l'hypothèse que la combinaison de différentes représentations vectorielles peut améliorer la qualité du score de similarité calculé, si ces représentations sont suffisamment complémentaires. Plus concrètement, nous pensons qu'une variation des paramètres de prétraitement des corpus et des paramètres de construction des vecteurs peut diversifier la représentation sémantique, et que l'obtention des représentations les plus complémentaires est en mesure d'orienter les calculs de similarité sur différents aspects sémantiques. Les paramètres pris en compte dans notre système sont les suivants :

size (entre 2 et 10000) indique la taille des vecteurs générés. Une même phrase représentée sur 50 dimensions ou 3000 dimensions portera des indices sémantiques différents, dans la granularité de ces indices (aspects topicals ou aspects sémantiques plus fins).

removePunct (vrai ou faux) indique la suppression ou non de la ponctuation de chaque phrase. La structure de la phrase variera selon la valeur de ce paramètre. Une virgule indiquera une séparation entre deux parties de phrase. Un point d'interrogation indiquera que la phrase est une interrogation ou une demande, et que les informations énoncées ne sont probablement pas factuelles. En revanche, supprimer toute ponctuation produira des vecteurs ne se focalisant que sur les mots et leurs voisins.

window (entre 1 et 20) définit la taille de la fenêtre de contexte de chaque mot. Plus la fenêtre est grande, plus le contexte d'un mot sera constitué de voisins éloignés dans la phrase, à gauche comme à droite. Par exemple, dans la phrase *"Bob joue du piano"*, une *window* de 1 (qui correspondra à une fenêtre de 3 mots, sauf au bord des phrases)

centré sur *Bob* prendra en compte l'action (le verbe *jouer*) alors qu'une *window* de 4 considérera aussi sur quoi s'applique l'action (le *piano*).

toLowerCase (vrai ou faux) indique la conservation ou non des majuscules présentes dans la phrase. Les majuscules peuvent, par exemple, permettre de différencier le nom commun d'un nom personnel. Leur suppression permet de réduire les noms personnels à leur forme commune si celle-ci existe et permet également de ne pas différencier un même mot en début de phrase ou non.

removeStopWords (vrai ou faux) indique si les mots vides sont supprimés de chaque phrase ou non. Les mots vides sont des mots peu informatifs mais qui permettent de lier les mots informatifs et structurer la phrase. Leur suppression permettra de focaliser l'analyse sur les mots sémantiquement riches. Au contraire, les conserver permettra de mieux représenter l'enchaînement des mots sémantiquement riches.

lemma (vrai ou faux) indique si la phrase est lemmatisée ou non. Une phrase lemmatisée perd en information sémantique puisque, par exemple, en supprimant la conjugaison, le lien entre un verbe et son sujet est affaibli. Une lemmatisation rendra la taille du vocabulaire beaucoup moins grande et les phrases seront plus proches, ce qui accentuera leur similarité « thématique ».

D'autres paramètres, directement liés à la génération des vecteurs par l'algorithme utilisé, ont aussi subi des variations : *alpha*, *iter*, *sample*, *negative* et *min_count*. Le lien entre certains hyper-paramètres liés à la constitution des vecteurs et le résultat final n'est pas simple à déterminer. Nous proposons donc une méthode d'optimisation qui cherche les combinaisons de paramètres les plus complémentaires.

La figure 1 schématise notre méthode en trois étapes fondamentales. Dans un premier temps l'optimisation des paramètres qui permet d'obtenir un ensemble de modèles classés selon leur performance avec la variation de paramètres décrite en sous-section 3.2. Conjointement à cette optimisation, nous générons des modèles avec des paramètres choisis au hasard. Ensuite, l'algorithme de sélection *topdesc* associe des modèles suffisamment différents en parcourant le classement issu de l'étape 1. Enfin, l'algorithme de sélection *topdelta* analyse les séries obtenues à l'étape 2 et associe les modèles les plus complémentaires, qui présentent une forte plus-value dans l'ensemble des séries. Nous mettons à disposition le code source python via la plateforme *GitHub* [1].

4 Combinaison de modèles complémentaires

4.1 Optimisation des paramètres

Nous commençons par générer des modèles [2] en cherchant les paramètres optimaux par une méthode d'optimisation de type recherche locale. Les onze paramètres cités en sous-section 3.2 ont été utilisés pour chaque modèle. Un modèle est généré grâce à l'outil *Doc2Vec*. *Doc2Vec*

1. `https://github.com/hayj/STSSeries/`.

2. Nous définissons un modèle comme étant l'ensemble des représentations vectorielles des phrases générées par l'outil *Doc2Vec* et ayant les mêmes paramètres de prétraitement de corpus (e.g. lemmatisation, mots vides) et de construction des vecteurs (e.g. nombre de dimensions, taille de fenêtre).

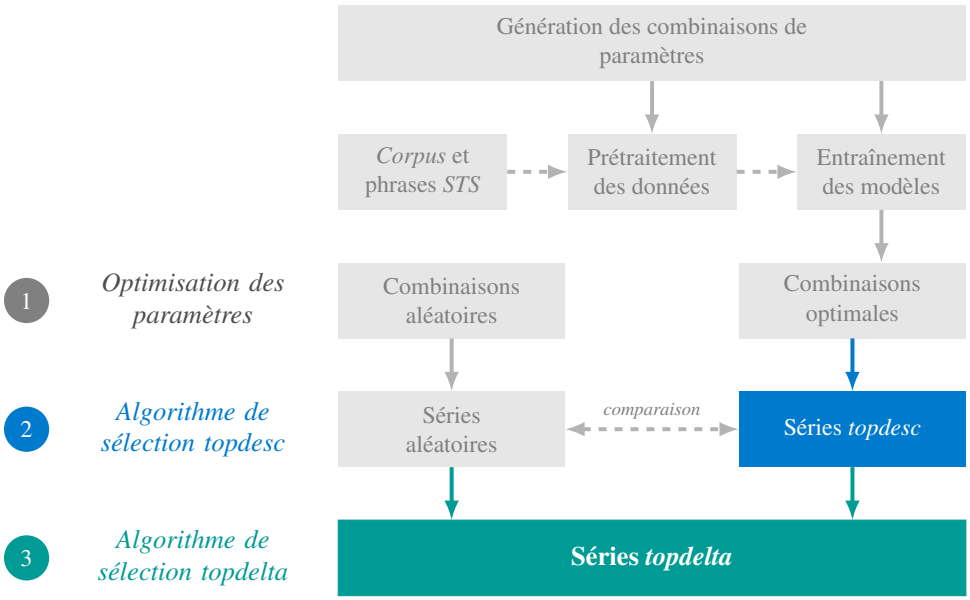

FIG. 1 – *Schéma illustrant les trois étapes : optimisation, algorithmes de sélection topdesc et topdelta*

prend un corpus (un ensemble de phrases) prétraité par une partie des paramètres, et prend également des paramètres propres à l'exécution de l'outil. Cet outil donne en sortie, pour chaque phrase, un vecteur représentatif. La similarité de chaque de paire phrases peut donc être calculée grâce à ces vecteurs. D'autres modèles sont générés avec affectations de paramètres aléatoires. Plus de 50 000 modèles ont été générés lors de cette première étape : la moitié par la procédure d'optimisation et l'autre moitié grâce à une affectation aléatoire des onze paramètres. Chaque modèle permet d'obtenir des représentations vectorielles différentes de toutes les phrases *STS*. Les représentations vectorielles sont apprises par *Doc2Vec* sur les données de la tâche *STS* ainsi que sur le *Brown Corpus* pour un total d'environ 85 000 phrases (dont environ 14 000 provenant de la tâche).

Dans l'objectif d'obtenir un score pour chaque modèle reflétant ses performances sur la tâche *STS*, nous intégrons chaque modèle dans le système *DLS 2015* proposé par Sultan et al. (2015). Le calcul de similarité sémantique est appris par une régression linéaire de type *Ridge*. Le système *DLS 2015* est constitué de deux descripteurs :

1. Le premier correspond à un score d'alignement entre deux phrases. Ce score est obtenu en fonction du nombre de mots que l'aligneur a réussi à relier entre les deux phrases selon différentes métriques (dictionnaire de synonymes, distance de Levenshtein, etc).
2. Le second descripteur correspond à une similarité cosinus reprenant les vecteurs de Baroni et al. (2014).

Lorsque nous intégrons un modèle, cela signifie que nous prenons chaque vecteur de chaque phrase (i.e. les représentations vectorielles issues du modèle entraîné avec une certaine

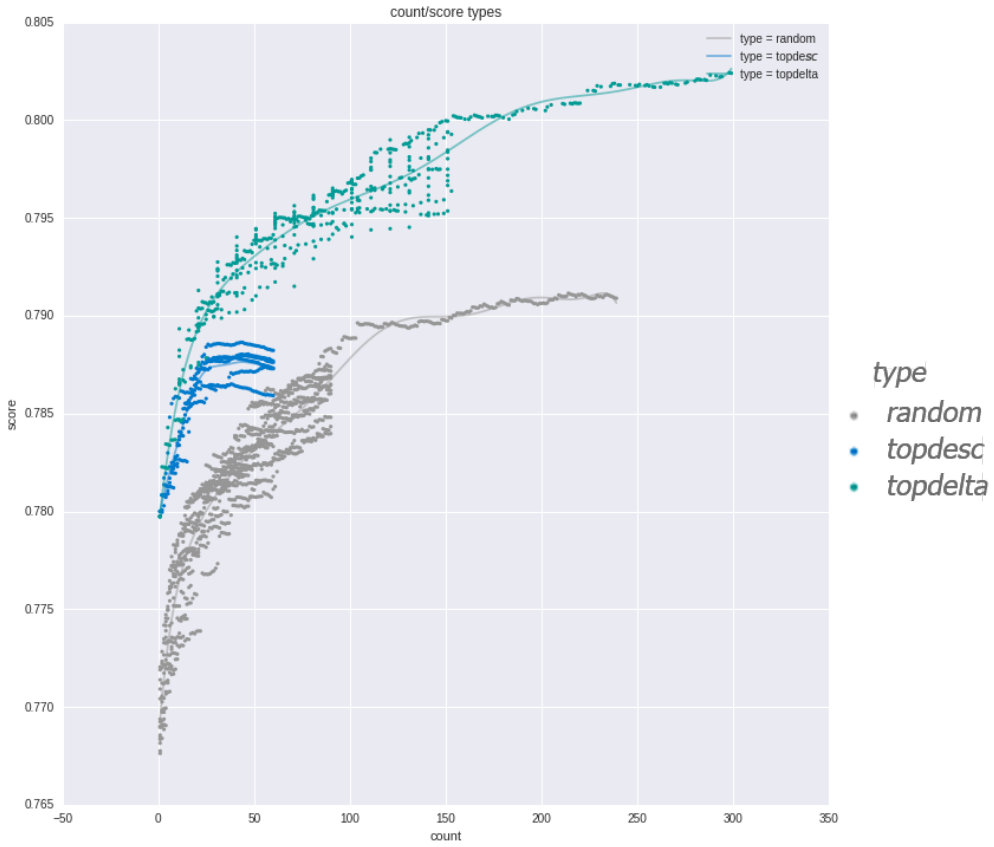

FIG. 2 – *Graphique rassemblant les trois types de séries*

combinaison de paramètres), puis nous calculons les similarités cosinus entre chaque paire de phrases. Ces similarités sont ajoutées en descripteur dans le système *DLS 2015*. Par la suite, l'utilisation de plusieurs modèles produira donc plusieurs descripteurs.

Pour notre première phase d'optimisation, nous n'utilisons qu'un seul modèle en descripteur additionnel. Nous évaluons un modèle (défini par son affectation de paramètres) par la performance du système *DLS 2015* avec, en plus, le descripteur de ce modèle. Enfin, chaque modèle est classé selon ses performances par évaluation croisée sur toutes les données antérieures à celles de l'année 2016.

4.2 Algorithme de sélection *topdesc*

Une fois ce classement obtenu, nous utilisons l'algorithme *topdesc* pour sélectionner des « séries » de modèles [3]. L'algorithme *topdesc* prend en entrée une « description », parcourt tous

3. Nous définissons une « série » de modèles comme étant un ensemble ordonné de modèles ayant différents paramètres de prétraitement de corpus et de construction des vecteurs.

```
[ (["size", "windows"], [100, 2], 20),
  (["removeStopWords", "removePunct", "lemma"], None, 6),
  (["min_count", "negative"], [10, 2], 10) ]
```

FIG. 3 – *Exemple de description d'une série topdesc*

les modèles générés lors de la phase d'optimisation dans l'ordre (du plus performant au moins performant) et sélectionne itérativement ou ignore les modèles. Il retourne en sortie une série de modèles correspondant à la description donnée. L'algorithme sélectionne les modèles sur deux critères :

— le modèle sélectionné doit être suffisamment différent des autres ;
— celui-ci doit être performant lorsqu'il est utilisé seul.

La description de la série se compose de plusieurs éléments :

1. les paramètres à différencier ;
2. la différence minimum avec les modèles déjà sélectionnés pour chaque paramètre à différencier ;
3. le nombre de modèles à sélectionner.

La figure 3 montre un exemple de description. La première ligne signifie que l'algorithme doit sélectionner 20 modèles avec des *size* et *window* variant respectivement d'au moins 100 et 2, ensuite 6 modèles variant sur quelques paramètres de prétraitement, et ainsi de suite. Tous les autres paramètres qui n'entrent pas dans la « différenciation » seront les paramètres optimaux trouvés lors de la phase d'optimisation.

La figure 2 montre en bleu l'ensemble des scores pour différentes descriptions et en gris les séries générées aléatoirement. Chaque point correspond au score de la série en fonction d'un certain nombre de modèles utilisés dans la série. Les points situés à droite associent donc plus de modèles que les points plus à gauche, ce qui permet d'observer l'évolution des performances de la série.

L'abscisse correspond donc au nombre de modèles sélectionnés et l'ordonnée correspond au score du système *DLS 2015* comprenant, en descripteurs additionnels, les similarités cosinus de tous les modèles de la série courante U_i. Chaque point du graphique appartient à une courbe $U_i(x)$, x étant un nombre de similarités cosinus (descripteurs) provenant des modèles de la série. Le score en ordonnée correspond donc au système *DLS 2015* avec un nombre x de descripteurs additionnels appartenant à la série U_i. Les séries se prolongent en $x + 1$, le score correspondra alors au même système, i.e. aux mêmes descripteurs, composé de la similarité d'un modèle en plus sélectionné pour la série U_i.

L'objectif de cette seconde étape était d'observer les modèles et leur pouvoir d'amélioration dans différentes séries. Une dizaine de séries ont été testées en faisant varier des descriptions semblables à la figure 3. Pour des raisons de lisibilité, seulement 5 d'entres elles ont été représentées sur la figure 2.

La sélection *topdesc* permet un gain d'environ 1.8% sur la base du système état de l'art 2015 (*DLS 2015*) et du meilleur modèle toujours présent en début de chaque série. Nous remarquerons que les séries générées aléatoirement permettent de surpasser les séries *topdesc* si suffisamment de modèles sont utilisés. Pour contrebalancer cet effet, nous introduisons l'algorithme *topdelta*.

4.3 Algorithme de sélection *topdelta*

L'algorithme de sélection *topdesc* permet de créer des séries de modèles performants. Mais comme nous avons pu le constater, il ne permet pas un gain important en performance. Un algorithme cherchant les modèles les plus complémentaires doit être capable de discriminer automatiquement les paramètres qui influent le moins sur la diversité des représentations. Par exemple, nous pouvons supposer que certains paramètres comme le nombre d'itérations lors de la construction des vecteurs sont directement liés aux performances du modèle et qu'une variation de ces paramètres ne permettra pas nécessairement d'orienter la représentation sur des aspects sémantiques variés. La sélection ne se faisait que sur le critère de performance et ne prenait pas en compte l'apport concret du calcul de similarité issu du modèle parmi l'ensemble des descripteurs. L'algorithme *topdelta* permet quant à lui de combiner des modèles par leur pouvoir de complémentarité. L'hypothèse sous-jacente est que les modèles les plus « complémentaires » sont ceux améliorant le plus une série. Cet algorithme consiste en l'attribution d'un score de « potentiel de complémentarité » à chaque modèle utilisé dans les séries *topdesc* et aléatoires. Une série *topdelta* correspondra donc à la suite de modèles ayant les meilleurs scores. Plusieurs séries différentes peuvent être générées en fonction des affectations de paramètres de l'équation calculant le score de « potentiel » que nous allons détailler.

Plus concrètement, nous pouvons intuitivement considérer que les modèles les plus complémentaires pouvant faire partie d'une série tiennent compte de deux facteurs :

1. d'une part la performance du modèle seul, i.e. son score en dehors d'une série ;
2. d'autre part son « pouvoir de complémentarité », i.e. la différence moyenne d'amélioration des performances entre ce modèle et l'ensemble de tous les modèles prédécesseurs possibles (i.e. qui apparaissent avant dans une série).

Le facteur score seul correspond aux scores obtenus lors de notre première étape. Le deuxième facteur est plus compliqué à mesurer. En effet, le pouvoir de complémentarité moyen d'un modèle doit tenir compte, dans l'idéal, de toutes les associations possibles de celui-ci et de prédécesseurs puisqu'un modèle peut améliorer une série uniquement grâce aux mauvaises performances de ses prédécesseurs. Chaque modèle est utilisé en moyenne trois fois et correspond donc en moyenne à trois points sur la figure 2. Il a donc été possible de faire une moyenne des différences entre le score de la série et le score de la série combiné au modèle courant.

Le score *topdelta* est défini ainsi :

$$potentiel(x) = (1-\alpha)S + \alpha \frac{\sum_{i=1}^{n} \frac{\tau_i(x)+\Delta_i}{1+\beta}}{n} \quad (1)$$

Le paramètre α permet de régler la balance entre l'influence du score seul S et celle de la complémentarité Δ. La complémentarité Δ est la différence moyenne entre le modèle x courant auquel nous voulons attribuer un score et l'ensemble de tous les prédécesseurs existants dans l'ensemble des séries *topdesc* déjà générées. Le modèle x peut également être un modèle présent dans les séries générées aléatoirement. Le paramètre Δ est normalisé entre sa valeur minimale et maximale parmi toutes les séries. Pour plus de lisibilité, cette normalisation n'est pas incluse dans les équations.

Plus un modèle sera loin dans une série, plus faible sera la probabilité qu'il améliore cette série. Ce constat est général dans la plupart des tâches en apprentissage automatique : tout

descripteur peut être indépendamment très performant, mais les performances associées de plusieurs descripteurs ne correspondront pas à la somme de leurs performances individuelles. Il est donc pertinent d'introduire un « bonus » qui permet d'augmenter le *delta* en fonction de la position du modèle dans la série. Le bonus τ a donc été introduit dans l'équation 1 et normalisé par sa borne supérieure β. Ce bonus peut être défini selon deux informations :

1. le nombre de prédécesseurs, puisque plus le modèle a de prédécesseurs, moins il a de chance d'améliorer la série ;
2. le score de la série au niveau de ce modèle, puisque plus le score de la série est élevé par rapport aux autres séries, moins ce modèle aura de chance d'améliorer globalement la série. Afin de simplifier les équations, cette information sera considérée comme normalisée au même titre que Δ et S.

Le bonus τ du modèle x pour la série courante i correspond donc à l'équation 2. Avec $nbAncestors$ le nombre de prédécesseurs, σ réglant l'influence des deux informations et β réglant l'importance du bonus τ (i.e. plus β sera grand, plus le bonus pourra être grand) :

$$\tau_i(x) = \sigma(\beta \times \frac{nbAncestors(x,i)}{nbAncestorsMax}) + (1-\sigma)(\beta \times S_{serie_i}) \quad (2)$$

La figure 2 montre les séries *topdelta* générées et optimisées (par une méthode de type recherche locale) sur les paramètres α, β, σ. Les meilleures affectations, représentées par la courbe *topdelta* la plus haute sur le graphique, sont $\alpha = 0.9$, $\beta = 1.0$, $\sigma = 0.5$. Nous remarquerons que le facteur de complémentarité est plus important que le score du modèle seul comme le montre l'affectation de α. L'affectation de β montre que le bonus est aussi important que *delta* lui-même. Enfin, l'affectation de σ ne priorise pas l'information du nombre de prédécesseurs par rapport au score de la série.

Sur la figure 2, pour des raisons de complexité en espace mémoire, nous avons prolongé uniquement une série *topdelta* et une série aléatoire, en choisissant la meilleure parmi celles déjà générées. Si l'on considère le score de référence comme étant le système *DLS 2015* avec un modèle généré aléatoirement, alors la série *topdelta* permet un gain de 4%. La prochaine partie est destinée à tester expérimentalement la meilleure série *topdelta* en évaluant notre système sur les données de test mises à disposition en 2016.

5 Expérimentations

Lors de la campagne *SemEval* de 2016 (Agirre et al., 2016), ce sont 43 équipes qui ont participé à la tâche pour un total de 119 essais. Les scores globaux allaient de 0.4 à 0.77 avec une médiane à 0.69. La *baseline* de la campagne 2016 correspond à une similarité basée sur la similarité cosinus des représentations « sac-de-mots » des paires de phrases et a obtenu un score de 0.51.

Le tableau 1 montre que la meilleure série *topdelta* a effectivement amélioré significativement les performances du système *DLS 2015*. Grâce à la sélection automatique de modèles complémentaires, notre système a pu obtenir un score au dessus de la médiane des scores de la campagne de 2016.

Système	Score
Baseline	0.51334
DLS 2015	0.69797
DLS 2015 + TopdeltaSerie	**0.73408**

TAB. 1 – *Score et comparaison*

6 Conclusion et perspectives

À travers notre travail expérimental sur la recherche de complémentarité, nous avons pu montrer qu'il est possible de sélectionner des représentations vectorielles suffisamment complémentaires pour guider le calcul de similarité sur différents aspects sémantiques. Cependant, pour des raisons de temps de calcul, nous n'avons entraîné nos modèles que sur un corpus restreint composé du *Brown Corpus* et des données *SemEval* antérieures à 2016.

Par la suite, nous pensons améliorer notre méthodologie pour qu'elle puisse s'appliquer à de plus larges corpus, ce qui permettrait de prendre en compte d'autres paramètres à faire varier comme le ciblage des entités nommées, que l'on peut conserver ou non dans les phrases. Les entités nommées forment un vocabulaire très large et nous pensons qu'elles jouent un rôle particulier en similarité sémantique textuelle.

Les méthodes supervisées qui ont montré obtenir de meilleurs résultats (Rychalska et al., 2016) sur la tâche *STS* pourront aussi être utilisées dans notre recherche de complémentarité. De plus, d'autres méthodes de représentation vectorielle pourront être combinées à *Doc2Vec* dans l'objectif de capturer des aspects sémantiques différents et potentiellement complémentaires.

Références

Agirre, E., C. Banea, D. Cer, M. Diab, A. Gonzalez-Agirre, R. Mihalcea, G. Rigau, et J. Wiebe (2016). Semeval-2016 task 1 : Semantic textual similarity, monolingual and cross-lingual evaluation. In *Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval-2016)*, San Diego, California, pp. 497–511. Association for Computational Linguistics.

Baroni, M., G. Dinu, et G. Kruszewski (2014). Don't count, predict ! A systematic comparison of context-counting vs. context-predicting semantic vectors. *Acl*, 238–247.

Baroni, M. et A. Lenci (2010). Distributional memory : A general framework for corpus-based semantics. *Computational Linguistics 36*(4), 673–721.

Curran, J. R. (2004). *From distributional to semantic similarity*. Ph. D. thesis, University of Edinburgh, UK.

Harris, Z. (1954). Distributional structure. *Word 10*(23), 146–162.

Le, Q. V. et T. Mikolov (2014). Distributed representations of sentences and documents. In *Proceedings of the 31th International Conference on Machine Learning, ICML 2014, Beijing, China, 21-26 June 2014*, pp. 1188–1196.

Mikolov, T., K. Chen, G. Corrado, et J. Dean (2013). Efficient estimation of word representations in vector space. In *In Proceedsings of Workshop at ICLR*.

Mitchell, J. et M. Lapata (2008). Vector-based models of semantic composition. In *ACL 2008, Proceedings of the 46th Annual Meeting of the Association for Computational Linguistics, June 15-20, 2008, Columbus, Ohio, USA*, pp. 236–244.

Pennington, J., R. Socher, et C. D. Manning (2014). Glove : Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing, EMNLP 2014, October 25-29, 2014, Doha, Qatar, A meeting of SIGDAT, a Special Interest Group of the ACL*, pp. 1532–1543.
English

Řehůřek, R. et P. Sojka (2010). Software Framework for Topic Modelling with Large Corpora. In *Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks*, Valletta, Malta, pp. 45–50. ELRA. `http://is.muni.cz/publication/884893/en`.

Rychalska, B., K. Pakulska, K. Chodorowska, W. Walczak, et P. Andruszkiewicz (2016). Samsung poland nlp team at semeval-2016 task 1 : Necessity for diversity ; combining recursive autoencoders, wordnet and ensemble methods to measure semantic similarity. In *Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval-2016)*, San Diego, California, pp. 602–608. Association for Computational Linguistics.

Sultan, M. A., S. Bethard, et T. Sumner (2015). Dls@cu : Sentence similarity from word alignment and semantic vector composition. In *Proceedings of the 9th International Workshop on Semantic Evaluation (SemEval 2015)*, Denver, Colorado, pp. 148–153. Association for Computational Linguistics.

Turney, P. et P. Pantel (2010). From frequency to meaning : Vector space models of semantics. *Journal of artificial intelligence research 37*(1), 141–188.

Van de Cruys, T., T. Poibeau, et A. Korhonen (2013). A tensor-based factorization model of semantic compositionality. In *Conference of the North American Chapter of the Association of Computational Linguistics (HTL-NAACL)*, pp. 1142–1151.

Summary

The goal of the Semantic Textual Similarity task is to automatically quantify the semantic similarity of two text snippets. Since 2012, the task has been organized on a yearly basis as a part of the SemEval evaluation campaign. This paper presents a method that aims to combine different sentence-based vector representations in order to improve the computation of semantic similarity values. Our hypothesis is that such a combination of different representations allows us to pinpoint different semantic aspects, which improves the accuracy of similarity computations. The method's main difficulty lies in the selection of the most complementary representations, for which we present an optimization method. Our final system is based on the winning system of the 2015 evaluation campaign, augmented with the complementary vector representations selected by our optimization method. We equally present evaluation results on the data set of the 2016 campaign, which confirms the benefit of our method.