Évaluation de la similarité sémantique entre des phrases

El-Asmar, Jad **Elakhrass, Mohamed Ali** **Hasrouni, Elio**

Abstract

Dans ce document, la similarité sémantique sera calculée sur une paire de phrases afin de mesurer la ressemblance entre ces deux dernières, grâce à différentes techniques de NLP. La similarité sera évaluée sur une échelle allant de 0 (phrases ne possédant aucune similarité) à 5 (phrases équivalentes) sur un ensemble de données du STS Benchmark. Plusieurs algorithmes seront présentés pour le problème de l'évaluation de la similarité sémantique entre des phrases. Les résultats obtenus suite à différents baselines et modèles les plus performants seront présentés sous forme de graphiques afin de les comparer entre eux.

1 Introduction

La mesure précise de similarité sémantique entre des phrases est un défi que les chercheurs tentent de résoudre grâce à différentes techniques de NLP. Un des défis que ces derniers rencontrent est le manque de données d'entrainement. En effet, il est difficile d'obtenir un modèle qui apprend efficacement lorsqu'un petit nombre de données d'entrainement est fourni. Il est donc important de pouvoir obtenir une précision élevée lors de la mesure de similarité sémantique, et ce, même si peu de données d'entrainement sont fournies. Afin de résoudre ce problème, une approche de base, soit un TFIDF avec une similarité cosine, a été utilisée. La corrélation de Spearman obtenue était de 79.041%. Afin d'obtenir les meilleures performances pour accomplir cette tâche, des modèles peaufinés et pré-entrainés sur des milliards de données seront utilisés. Google s'est penché sur cette problématique et ont développé plusieurs modèles entrainés sur des milliards de données comme BERT et Universal Sentence Encoder. Facebook ont aussi développé fasttext, un modèle similaire. La solution choisie a été d'utiliser BERT, un encodeur bidirectionnel du type Transformeur pré-entrainé sur des milliards de données de Wikipédia. Avec ce modèle, le score de Spearman obtenu sur l'ensemble de test était de 88.759%, soit le meilleur résultat.

2 Méthodologie

2.1 Définition de la tâche/objectif

Le problème abordé est celui de mesurer précisément la similarité sémantique entre deux phrases. Deux phrases sont passées en entrée à un modèle avec leur score de similarité allant de 0 à 5. Plus le score se rapproche de 5, plus les phrases se ressemblent, et vice versa. L'objectif est de prédire la similarité des paires de phrases des données de test. Ainsi, il est important de pouvoir effectuer des prédictions se rapprochant le plus possible des vraies valeurs, afin d'obtenir un score de Spearman le plus élevé possible.

2.2 Définition de l'algorithme/méthode/technique

Le modèle qui donne le meilleur score de Spearman (soit de 88.759%), est basé sur BERT, un encodeur bidirectionnel de type Transformeur, qui est pré-entrainé sur des milliards de données de Wikipédia. La librairie Sentence-Transformers va effectuer différentes étapes afin d'améliorer les performances d'un modèle BERT pour détecter la similarité sémantique. La première étape effectuée est le pooling sur les vecteurs des phrases "embedded", afin d'obtenir un vecteur de taille fixe. Pour effectuer ceci, il y a plusieurs stratégies disponibles. Pour la problématique abordée, le pooling de moyennes (mean) donne le meilleur résultat et fut utilisé. Ensuite, la prochaine étape consiste à peaufiner (fine-tune) le modèle sur les données du STS Benchmark et celles du NLI. Cette étape permet de mettre à jour les poids afin de donner une signification sémantique aux embeddings. Toujours dans cette étape, la similarité cosine est calculée entre les deux vecteurs en utilisant l'erreur MSE comme fonction objective. Le

modèle tente de réduire cette erreur d'une époque à l'autre. Le modèle obtenu est "bert-large-nli-stsb-mean-tokens".

```
function encode :
    entrée: liste de phrases
    sortie: liste de prongements de phrase

    for each phrase do
        tokenize(phrase)

    features = {}
    for each phrase tokenized do
        feature_phrase <-- obtenir les features

        for feature in feature_phrase do
            features[feature] <-- embedding du feature

    embeddings <-- forward(features)

    return embeddings
```

Figure 1: Pseudocode de la fonction encode

Afin d'encoder les phrases à l'aide du modèle décrit plus haut, il faut utiliser la fonction encode de la classe SentenceTransformers. Cette classe hérite de la classe Sequential de PyTorch. Cette fonction prend en entrée une liste de phrases et retourne en sortie la liste de phrases encodées. La fonction encode peut être résumée en cinq étapes. La première étape consiste à convertir la phrase en tokens à l'aide de BertTokenizer. BertTokenizer sépare les mots et la ponctuation. De plus, il ajoute le token cls au début. La deuxième étape effectuée par la fonction est l'extraction des features. Une feature correspond à un mot. Une fois les features extraits, la matrice phrase par features est créée. Cette matrice est passée en entrée dans la fonction forward de la librairie PyTorch. Cette fonction est surchargée par la classe SentenceTransformers. Son but est de sortir les calculs nécessaires pour le embedding de chaque phrase. Finalement, une fois les embeddings des phrases obtenus, la métrique utilisée pour calculer la similarité entre deux phrases est la distance cosine. Afin, d'obtenir cette distance, la fonction cosine de librairie scipy est utilisée.

3 Évaluation expérimentale

3.1 Ensemble de données, métriques et baselines

Les données utilisées sont ceux du STS-Benchmark. Il y a 5703 paires de phrases dans les données d'entrainement. Elles proviennent de légendes d'images, de titres d'actualités et de forums. La diversité de la provenance de ces phrases reflètent leur réalité.

L'hypothèse initiale obtenue en observant la quantité de données disponibles est que les modèles pré-entrainés obtiendront une meilleure corrélation de Spearman que les modèles entrainés sur les données à disposition. D'autres hypothèses découlent de celle-ci. La métrique la plus convenable pour obtenir la similarité entre deux phrases serait la distance cosine. Le type de pooling serait le mean et la version large de BERT serait la plus avantageuse. La corrélation de Spearman est employée tout au long des expériences et tests réalisés. Cette métrique ne prend pas en compte les valeurs prises, mais les rangs de celles-ci.

Les variables indépendantes du modèle utilisé sont la mesure de similarité employées, le type de BERT (base ou large) et le type de pooling choisi. Dans ce cas, trois métriques pour calculer la similarité entre deux plongements de phrases ont été testées avec les données STS Benchmark. En se servant de la corrélation de Spearman, les résultats suivants sont obtenus pour la mesure cosine, euclidienne et manhattan sont respectivement de 88.759% , 88.519% et 88.486% .

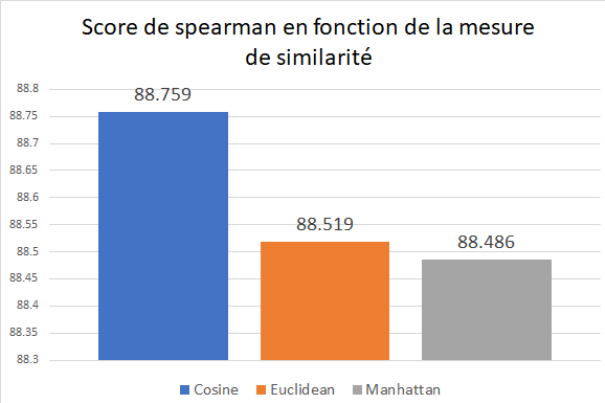

Figure 2: Corrélation de Spearman en fonction de la métrique de mersure de similarité utilisée

Maintenant, qu'il est évident que la mesure cosine est la plus avantageuse pour les données STS, elle sera employée pour les prochains tests. La prochaine variable indépendante qui sera expliquée est le type de pooling. Effectivement, trois exemples de pooling peuvent être choisis. Le pooling MEAN permet d'avoir une représentation vectorielle des phrases en effectuant la moyenne des vecteurs de mots obtenus des plongements de BERT. Cela fait en sorte que chaque phrase aura X moyennes dans son vecteur ou X représente le

nombre de tokens dans la phrase. En ce qui concerne le pooling MAX, celui-ci représente chaque phrase par la valeur maximale de chaque vecteur des mots de la phrase. Le dernier type de pooling est le pooling CLS qui représente chaque phrase par la première valeur des vecteurs de mots. Les résultats obtenus sont 86.62% pour CLS, 69.92% pour MAX et 87.44% pour MEAN. Le pooling choisi sera alors MEAN.

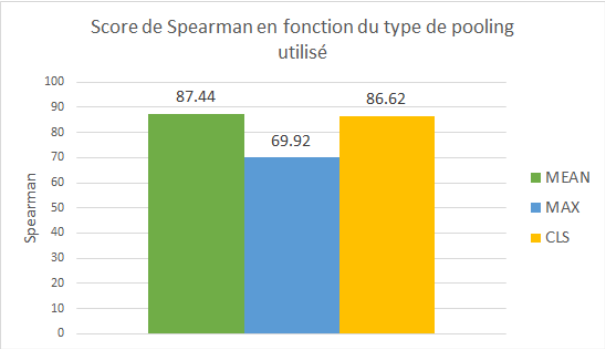

Figure 3: Correlation de Spearman en fonction du type de pooling utilisé

La dernière variable indépendante est le type de BERT utilisé. BERT base contient uniquement 12 couches d'encodeurs, donc 110 millions de paramètres. Quant à lui, BERT large contient 24 couches d'encodeurs et 340 millions de paramètres. Dans la problématique courante, la version large est plus avantageuse. En effet, BERT large permet d'obtenir 88.77% et BERT base 88.33%. Pour commencer à diagnostiquer cette problématique, un baseline a été développé. Il consistait à un embedding de phrases avec une métrique pour calculer la similarité. Pour l'embedding des phrases, plusieurs modèles ont été utilisés tels que Glove, Word2Vec, USE, ajout de smooth inverse frequency... Plusieurs métriques ont été utilisés telles que Word mover distance, la distance cosine, la distance de Manhattan, ... Toutes ces combinaisons ont été testées sur le même ensemble de test et en utilisant la corrélation de Spearman. La combinaison gagnante était une vectorisation TF-IDF pour chaque phrase et une mesure de similarité cosine.

Afin d'améliorer un peu le baseline, des étapes de prétraitement ont été employées pour aboutir à la combinaison la plus convenable. Au bout du compte, le pipeline pour le prétraitement consiste en un retrait des majuscules et de la ponctuation, une tokenisation des phrases, la lemmatisation des tokens et un retrait des stopwords. La tokenisation utilise la méthode word_tokenize de la librairie NLTK. La lemmatisation utilise WordNetLemmatizer de NLTK en taguant chaque token en se servant de la fonction pos_tag, afin de taguer les mots étant des noms, verbes, adverbes ou adjectifs. Dans le but de retirer les stopwords, la collection de stopwords anglais de la librairie NLTK a été employée.

Pour aller plus loin, plusieurs réseaux neuronaux ont été implémentés. Cependant, vu la quantité de données limitée disponible, il a été posé comme hypothèse que le score serait inférieur au baseline. Afin de palier à ce problème, il a été nécessaire de créer des données. Une fois les données d'entrainement séparées en données d'entrainement et de validation à l'aide de la fonction train_test_split de la librairie sklearn, il a été possible de doubler nos données d'entrainement en permutant les phrases. En effet, supposons que la paire phrase1 et phrase2 donne un score, la paire phrase2 et phrase1 donnera le même score. Le réseau de neurones implémenté obtenant un score acceptable est un réseau de neurones siamois, c'est-à-dire qu'il a deux entrées qui partage les mêmes couches. La première couche est une couche Embedding, dont la matrice de vecteurs utilise les valeurs du modèle glove.6b.300. Ce modèle est entrainé sur les données de Wikipedia et de Gigawords. Ensuite, les deux entrées sont concaténées. Finalement, pour la sortie, il y a une couche Softmax. Afin d'obtenir de meilleurs résultats, le score de 0 à 5 est classifié en 6 classes. Par exemple, un score de 4.5 a 50% de chance d'être de la classe 4 et 50% de chance d'être de la classe 5. Ce modèle nous a donné un score de 75.411%.

Après quelques recherches et lectures d'articles, il était rendu évident que les modèles les plus performants seraient les modèles pré-entrainés. Comme expliqué dans la méthodologie, un modèle pré-entrainé de BERT fine-tuned sur les données NLI et STS était le plus avantageux.

3.2 Résultats

En ce qui concerne la baseline initiale, celle-ci obtenait un score d'environ 74.973%. Dans certaines baselines, le prétraitement diminuait la corrélation, mais en ce qui concerne la baseline choisie, celle-ci est avantagée. Avec l'ajout des étapes de prétraitement, la corrélation est devenue 79.041%. Une amélioration significative

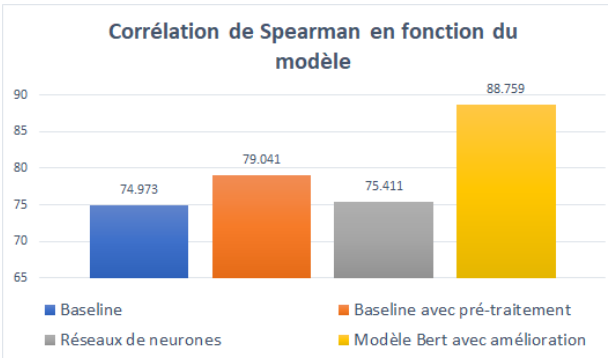

Figure 4: Corrélation de Spearman en fonction du modèle utilisé

d'environ 5% est visible. Pour les modèles employant des réseaux de neurones,le meilleur score obtenu était de 75.411%. Ce score est inférieure au baseline. Finalement, la dernière approche analysée, qui était l'utilisation de BERT, a donné de biens meilleurs résultats. Effectivement, en observant le graphique, la corrélation de Spearman obtenue est de 88.759%, soit une amélioration d'environ 10% de la baseline choisie.

3.3 Discussion

Selon les résultats obtenus plus haut, les hypothèses sont vérifiés. En effet, l'utilisation d'un modèle de langue tel que BERT donne de bien meilleur résultat. Évidemment, puisque celui-ci a été pré-entrainé sur des milliards de données Wikipédia et a été peaufiné (fine-tuned) sur les données STS benchmark, sa corrélation de Spearman sera supérieure. Étant entrainé pendant de nombreuses heures à réduire l'erreur MSE sur la mesure de similarité cosine entre des paires de phrases de cet ensemble de données représente un énorme avantage. De plus, l'utilisation d'un réseau de neurones ne permet pas de surpasser le score du baseline. En effet, tel qu'expliqué plus haut, ce modèle est limité par la quantité de données disponible. L'hypothèse a été vérifiée.

4 Travaux connexes

Plusieurs modèles ont été essayés pour être testé sur les données STS benchmark et dans la même problématique, soit de calculer la similarité entre deux phrases. Trois grosses catégories de modèles ont été survolées. Des modèles non entrainés sur les données STS benchmark tels que word2vec avec smooth inverse frequency (SIF), word mover distance (WMD) avec Glove et plusieurs autres. La deuxième catégorie contient les modèles pré-entrainés sur les données STS uniquement. La troisième catégorie contient des modèles adaptés aux données STS et NLI. Le modèle avec lesquelles les comparaisons suivantes auront lieu est celui de BERT avec les améliorations décrites plus haut adaptées sur les ensembles de données STS et NLI.

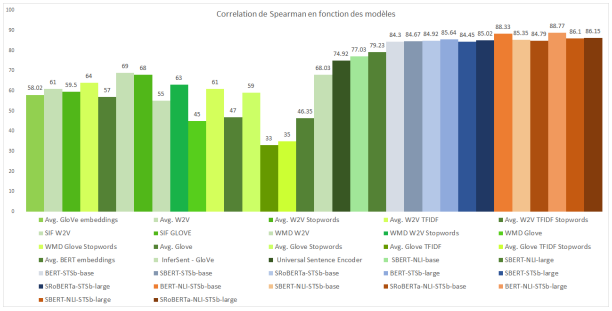

Figure 5: Corrélation de Spearman en fonction du modèle utilisé

En observant le graphique, plusieurs informations peuvent être ressorties. En effet, les scores des modèles qui ne sont pas pré-entrainés sur les données STS possèdent majoritairement des scores faibles. La majorité de ces modèles utilisent Word2vec ou Glove qui se basent sur des statistiques de co-occurrence. Contrairement à BERT, ces deux techniques ne prennent pas en compte le contexte de chaque phrase ce qui explique leur score plus faible. Pour ce qui est des deux derniers modèles de la première catégorie (vert), ceux-ci sont pré-entrainés sur les données NLI uniquement. Leur score est de 77.03% et 79.23% soit plus faible d'environ 5% à 10% que les modèles entrainés sur STS ou STS et NLI. Les données NLI ne sont pas exactement similaires aux données STS. En effet, deux phrases n'ont pas un score entre 0 et 5 associé, mais une étiquette indiquant la relation entre les deux, par exemple contradiction. Malgré cette différence, ce modèle doit prendre en compte le contexte. Cela explique la corrélation de Spearman plus faible, mais plus élevée que les autres modèles n'utilisant pas BERT. InferSent utilisant Glove obtient 68.03% ce qui est environ 20% plus faible que BERT large utilisé. Une seule couche de Bi-LSTM est utilisée en comparaison avec 24 couches dans BERT. Beaucoup plus de paramètres sont générés ce qui donne l'impression de rendre le modèle plus performant. En général, tous les modèles de cette première catégorie n'effectuent pas de généralisation sur les données STS ce qui explique

leur corrélation inférieure.

La seconde catégorie contient les modèles utilisant BERT ou des dérivées de BERT en appliquant un fine-tuning sur les données STS seulement. De première vue, l'amélioration est visible. Les corrélations de Spearman varient entre 84.3% et 85.02%. Effectivement, dû au fine-tuning ces modèles ont pu trouver des relations et particularités sur l'ensemble de données en question.

Pour la dernière catégorie, il est intéressant de voir que les scores ont une amélioration significative par rapport aux modèles fine-tuner uniquement sur les données STS. Cela est compréhensible puisque plus de données sont passées, les contextes sont alors mieux interprétés et compris par le modèle. Puisque l'ensemble de données consiste aussi en une comparaison de deux phrases anglaises, la logique de la problématique est identique. En effet, les corrélations de Spearman varient entre 84.79% et 88.77%. D'ailleurs le modèle expliqué dans cet article représente celui ayant le meilleur score des trois catégories présentées ci-dessus.

5 Travaux futurs et conclusion

Les principales limites de l'algorithme utilisé sont les ressources consommées et le fait que BERT ait uniquement été entraîné sur des données de Wikipedia.

Pour la première limite, il serait possible d'utiliser une des dérivées de BERT tel que ALBERT. Ce dernier utilise environ 70% du nombre de paramètres employés par les couches de BERT. Cette différence provoque des gains de performances tout en consommant moins de ressources et une plus petite durée d'entrainement. Celui-ci obtiendrait une corrélation autour de 92.5% sur l'ensemble de données STS, selon quelques articles.

L'étape de pré-entrainement de BERT sur les données de Wikipedia pourrait le désavantager s'il était employé sur un ensemble de données différent. Effectivement, avec l'ensemble de données SICK-R, BERT est surpassé par le Universal Sentence Encoder (USE) de Google. La raison est la suivante, l'entrainement de ce modèle a eu lieu sur des millions de données variées tels que des nouvelles, forums de questions et réponses et des forums de discussions. Ayant été exposé à une si grande diversité de textes, ce modèle obtiendrait de meilleurs résultats dans d'autres ensembles de données que STS.

Grâce à ce défi, il est maintenant évident de voir que les modèles les plus performants dans le monde du NLP sont les gros modèles pré-entrainés comme BERT. La quantité de données est significative pour la performance d'un modèle. Dans plusieurs cas, les données ne sont pas aussi nombreuses et les modèles pré-entrainés sont présents pour combler ce besoin. Il est aussi intéressant de voir que la mesure de similarité la plus efficace était la distance cosine. Les expériences et résultats obtenus dans cet article montrent qu'il est possible d'obtenir une corrélation près de 90% avec les valeurs de l'analyse humaine. Dans les situations ou une quantité limitée de données est mise à disposition des chercheurs, ils n'auront pas à chercher trop loin et pourront directement commencer par utiliser des modèles pré-entrainés et essayer de les améliorer.

References

1- Reimers, N. and Gurevych, I. (2018). Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks. [en-ligne] Arxiv.org. Disponible à : https://arxiv.org/pdf/1908.10084.pdf [Accédé le 19 Nov. 2019].

2- Alammar, J. (2019). The Illustrated BERT, ELMo, and co. (How NLP Cracked Transfer Learning). [en-ligne] Jalammar.github.io. Disponible à : http://jalammar.github.io/illustrated-bert/ [Accédé le 21 Nov. 2019].

3- Peirsman, Y. (2018). Simple Sentence Similarity. [en-ligne] NLPTOWN. Disponible à : https://github.com/nlptown/nlp-notebooks/blob/master/Simple%20Sentence%20Similarity.ipynb [Accédé le 18 Nov. 2019].

4- Raffel, Colin et al. (2019). Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer. [en-ligne] Google. Disponible à https://arxiv.org/pdf/1910.10683v2.pdf [Accédé le 18 Nov. 2019].

5- UKPLab. (2019). Sentence Embeddings with BERT XLNet. [en-ligne] UKPLab. Disponible à : https://github.com/UKPLab/sentence-transformers [Accédé le 19 Nov. 2019].