Modèles de langue appliqués aux schémas Winograd français

Olga Seminck Vincent Segonne Pascal Amsili
Laboratoire de Linguistique Formelle (Université Paris Diderot & CNRS)
8 place Paul Ricœur, 75013 Paris, Fance
olga.seminck@cri-paris.org, vincent.segonne@gmail.com,
pascal.amsili@linguist.univ-paris-diderot.fr

RÉSUMÉ

Les schémas Winograd sont des problèmes de résolution d'anaphores conçus pour nécessiter un raisonnement sur des connaissances du monde. Par construction, ils sont insensibles à des statistiques simples (co-occurrences en corpus). Pourtant, aujourd'hui, les systèmes état de l'art pour l'anglais se basent sur des modèles de langue pour résoudre les schémas (Trinh & Le, 2018). Nous présentons dans cet article une étude visant à tester des modèles similaires sur les schémas en français. Cela nous conduit à revenir sur les métriques d'évaluation utilisées dans la communauté pour les schémas Winograd. Les performances que nous obtenons, surtout comparées à celles de Amsili & Seminck (2017b), suggèrent que l'approche par modèle de langue des schémas Winograd reste limitée, sans doute en partie à cause du fait que les modèles de langue encodent très difficilement le genre de raisonnement nécessaire à la résolution des schémas Winograd.

ABSTRACT

Language Models applied to French Winograd Schemas

Winograd schemas are anaphora resolution problems built in such a way that reasoning about world knowledge is necessary to solve them. As a consequence, they are meant to be insensitive to simple statistical methods (based on coocurrence in corpus). However, today's state of the art systems use language models to solve Winograd schemas (Trinh & Le, 2018). In this paper, we report a study that tests similar language models on French Winograd schemas. It leads us to reconsider the evaluation metrics commonly used in the community. The results that we obtain, especially when compared to those of Amsili & Seminck (2017b), suggest that language model approaches to Winograd schemas remain limited, probably because language models, no matter how powerful they are, have difficulties to encode the kind of reasoning that is necesssary to solve Winograd schemas.

MOTS-CLÉS : Schémas Winograd, résolution d'anaphores, modèle de langue, context2vec, LSTM.

KEYWORDS: Winograd schemas, Anaphora Resolution, language model, Context2Vec, LSTM.

1 Introduction

Les schémas Winograd, proposés par Levesque *et al.* (2012) comme un test d'intelligence artificielle, sont des problèmes de résolution anaphorique conçus pour nécessiter un raisonnement sur des connaissances du monde. Chaque schéma est constitué d'une paire d'*items* : deux discours identiques à un mot (ou une expression) près, et qui comprennent une expression anaphorique à résoudre, dont l'antécédent change d'une version à l'autre. Ainsi, dans (1), la réponse naturelle pour la question

formée avec le mot faible (mot *spécial*) est *Nicolas* (R0), alors que la question formée avec le mot lourd (mot *alternant*) appelle la réponse *son fils* (R1).

(1) Nicolas n'a pas pu soulever son fils car **il** était trop ⟨faible/lourd⟩.
Qui était trop ⟨faible/lourd⟩ ? R0 : Nicolas
R1 : son fils

La collection de schémas en anglais publiée par Levesque *et al.* (2012) a fait l'objet de traductions dans d'autres langues : douze schémas ont été traduits en chinois, la collection entière a été traduite en japonais [1] et il existe une version de la collection traduite/adaptée en français à propos de laquelle il a été vérifié de plus que les schémas étaient 'Google-proofs' (non sensibles à des statistiques simples de co-occurrence de mots) et que les humains n'avaient pas de difficulté à les résoudre (Amsili & Seminck, 2017a,b).

Différentes approches ont été proposées pour résoudre les schémas Winograd, mais uniquement pour l'anglais. Nous présentons ici un premier système pour le français inspiré de méthodes à base de modèles de langue qui représentent actuellement l'état de l'art pour l'anglais (Trinh & Le, 2018) (exactitude de 63,7% sur 273 items Winograd). Ce résultat état de l'art est obtenu en combinant les scores de 14 modèles de langue distincts (qui utilisent différents algorithmes et corpus d'entraînement). N'ayant pas la possibilité d'en faire autant, nous présentons ici les résultats de deux modèles de langue : un modèle de langue récurrent et un modèle de contexte.

2 Méthode

2.1 Modèle de langue récurrent

Notre méthode est inspirée du travail de Trinh & Le (2018) ; elle consiste à générer, à partir de chaque item, une paire de phrases en remplaçant le pronom anaphorique par les antécédents possibles (2). Pour le français, ce remplacement demande parfois de ré-agencer la phrase, par exemple au moment de remplacer un pronom objet clitique (donc devant le verbe) par un groupe nominal plein (après le verbe). Nous avons toujours fait en sorte de ne pas présenter au modèle de langue des phrases agrammaticales. Après l'insertion des réponses, le modèle de langue permet d'attribuer une probabilité jointe à chaque phrase, l'hypothèse étant que celle qui obtient le meilleur score est celle qui contient la bonne réponse.

(2) a. Nicolas n'a pas pu soulever son fils car Nicolas était trop faible. (spe, R0)
b. Nicolas n'a pas pu soulever son fils car son fils était trop faible. (spe, R1)

Pour réaliser nos expériences, nous avons entraîné un modèle *Long Short Term Memory* (LSTM) (Hochreiter & Schmidhuber, 1997) sur la version française de Wikipedia [2]. Nous avons sélectionné les 100K mots les plus fréquents trouvés dans le corpus d'entraînement comme vocabulaire. Les

1. Les schémas chinois et japonais n'ont pas de publication associée. On peut les trouver sur la page : `https://cs.nyu.edu/davise/papers/WinogradSchemas/WS.html`. La traduction en chinois été faite par Wei Xu et la traduction en japonais par Soichiro Tanaka, Rafal Rzepka, et Shiho Katajima.

2. Nous avons utilisé la version fr-wikipedia 2016-12-12 construite par Coavoux (2017) avec l'outil de Giuseppe Attardi : `http://medialab.di.unipi.it/wiki/Wikipedia_Extractor`.

tailles des représentations vectorielles de mots et des couches cachées du réseau sont respectivement 1024 et 2048 et la minimisation de la perte a été réalisée avec l'algorithme *Adagrad* (Duchi *et al.*, 2011) avec un pas d'apprentissage de 0,2. Enfin un dropout de 0,25 a été appliqué sur la couche de sortie du LSTM.

Pour la résolution des schémas Winograd, l'application de cette méthode rencontre plusieurs types de problèmes et ne permet pas toujours au système de trancher entre les deux réponses possibles. Pour commencer, certains schémas sont atypiques dans le sens où il n'y a pas d'expression anaphorique que l'on puisse remplacer par les deux réponses possibles, voir par exemple (3) :

(3) Joël a vendu sa maison et en a acheté une nouvelle à quelques kilomètres.
Il va ⟨déménager/emménager⟩ ce jeudi.
Joël va ⟨déménager de/emménager dans⟩ quelle maison ?
R0 : son ancienne maison
R1 : sa nouvelle maison

Un autre problème vient du fait que les items Winograd peuvent être des discours constitués de plusieurs phrases (4), or le modèle de langue que nous utilisons n'est entraîné que sur des phrases isolées. On peut envisager plusieurs façons de gérer ce problème : soit concaténer les phrases, mais alors le modèle est confronté à des données assez différentes de ses données d'apprentissage, soit appliquer le modèle sur la dernière phrase, mais cette phrase seule ne contient en général pas les indices permettant la résolution, soit enfin entraîner des LSTM sur des discours de quelques phrases. Nous laissons de côté toutes ces solutions potentielles dans le présent travail, et décidons de ne pas répondre dans le cas d'un schéma multi-phrases.

(4) Fred est le seul homme encore vivant à se rappeler de mon arrière grand-père.
C' ⟨est/était⟩ un homme remarquable.
Qui ⟨est/était⟩ remarquable ? R0 : Fred
R1 : mon arrière grand-père

Enfin, même si les modèles LSTM sont capables de gérer le cas de mots inconnus d'une phrase (en les remplaçant par un token unique ⟨UNK⟩), il peut arriver que les réponses (R0 et R1) d'un schéma soient justement inconnues du modèle : c'est le cas en particulier des noms propres. Or, ce sont les seuls mots qui distinguent les deux variantes, qui obtiendraient donc dans ce cas le même score, empêchant notre système de trancher. Notre système produit donc soit la réponse correspondant à la variante ayant le meilleur score, soit une non-réponse dans les cas évoqués ci-dessus.

2.2 Modèle contextuel

Context2vec (C2V) (Melamud *et al.*, 2016) est un modèle qui permet de représenter le contexte d'un token par le biais de réseaux de neurones récurrents bi-directionnels. Ces représentations permettent de mesurer à quel point un token est attendu dans un contexte. Nous utilisons ce modèle afin d'évaluer laquelle des deux réponses correspond le mieux au contexte. Ce modèle est donc différent des modèles LSTM utilisés par Trinh & Le (et de celui de la section 2.1) : au lieu de considérer la probabilité conjointe de la phrase comme nous l'avons vu pour les modèles LSTM, nous mesurons la similarité entre un mot cible et son contexte. Le modèle C2V construit des représentations pour le contexte et

pour le token cible dans le même espace vectoriel, les rendant donc comparables.

Comme pour l'expérience avec le modèle LSTM, on crée deux versions de chaque item en remplaçant l'anaphore par les réponses et en ré-agençant la phrase afin d'éviter des phrases agrammaticales. Ensuite, le token cible (en jaune) est sélectionné ; tous les autres tokens de la phrase forment le contexte (en gris). Pour la résolution des schémas, on calcule donc la similarité entre une réponse et son contexte, on choisit celle qui obtient le plus haut score.

(5) a. Simon a expliqué sa théorie à Marc, mais **Simon** ne l'a pas comprise.

b. Simon a expliqué sa théorie à Marc, mais **Marc** ne l'a pas comprise.

Les réponses des schémas Winograd se composent souvent de plusieurs tokens. Comme il n'est pas possible de sélectionner plus d'un token comme cible, il faut donc choisir quel token de la réponse devient la cible. Nous appliquons pour cela l'heuristique suivante : la tête syntaxique de la réponse est choisie comme cible, et le reste de la réponse fait partie du contexte.

(6) La table ne passe pas par la porte parce que **la** **table** est trop large.

Il y a cependant trois exceptions à cette règle : si la tête syntaxique de la réponse est un mot fonctionnel, nous prenons la tête en-dessous de la tête (par exemple, si une réponse est 'du garçon', nous prenons le mot 'garçon' comme cible). La deuxième exception intervient lorsque les têtes syntaxiques ne permettent pas de distinguer entre R0 et R1 comme dans l'exemple (3) où la tête de R0 et R1 est 'maison'. Dans ce cas, on prend les mots distinctifs ; pour l'exemple (3), on prendra donc 'ancienne' et 'nouvelle' comme cibles. La dernière exception est le cas où il y a plusieurs candidats pour être la tête syntaxique (e.g. la réponse *Dr. Vincenot*). Dans ce cas, nous prenons le premier token comme la tête, c'est-à-dire *Dr*.

Nous avons utilisé l'implémentation de Melamud *et al.* (2016) pour entraîner le modèle C2V. Cet entraînement a été réalisé sur le même corpus Wikipedia que le modèle LSTM présenté précédemment. En ce qui concerne les paramètres du modèle, nous avons fixé à 300 la taille des représentations vectorielles construites et nous avons utilisé l'algorithme d'optimisation *Adagrad* (Duchi *et al.*, 2011) avec un pas d'apprentissage de $0,001$. Nous avons considéré deux méthodes de calcul de similarité : la similarité cosinus et une similarité qui consiste à appliquer un log sigmoïde au produit scalaire des deux vecteurs que nous comparons. Dans les deux cas, plus la similarité est grande, plus le mot cible est compatible avec le contexte.

Le modèle C2V bute sur le cas des mots inconnus, quand il s'agit des mots cibles : on ne peut pas comparer le vecteur d'un mot inconnu à une représentation vectorielle de contexte. Or, il existe beaucoup de schémas Winograd dans lesquels les réponses R0 et R1 sont des noms propres ou d'autres mots inconnus, ce qui nous conduit à un taux de non-réponse important.

3 Évaluation

La tâche que nous voulons évaluer consiste à déterminer pour chaque item quel est l'antécédent parmi les deux candidats. Il y a deux réponses possibles (et une seule correcte), la mesure la plus naturelle

est donc l'*exactitude*, qui mesure la proportion de bonnes réponses parmi les items : $\frac{h}{n}$, où h est le nombre de bonnes réponses et n le nombre total d'items dans la collection.

Cette mesure ne permet cependant pas de distinguer les réponses incorrectes et les non-réponses. Pour prendre en compte la performance du système dans les seuls cas où il répond, on peut introduire une mesure que nous pourrions appeler *qualité* : $\frac{h}{n-\theta}$, où θ désigne le nombre de non-réponses. La qualité est une mesure de précision : c'est la proportion de bonnes réponses rapportée au nombre total de réponses. Mais il est important de noter que nous ne sommes pas dans un cas où, parmi un ensemble d'items, il s'agit de décider lesquels entrent dans une catégorie donnée. Autrement dit, nous ne sommes pas dans le cas où on définit habituellement, à partir d'une table de confusion, le couple de mesures *précision* et *rappel*[3]. Il nous semble par conséquent plutôt inapproprié d'appeler *rappel* le rapport h/n (i.e. l'exactitude) comme le font Emami *et al.* (2018), et surtout de présenter la moyenne harmonique entre exactitude et qualité/précision (F1) comme une mesure synthétique pertinente. En effet ces deux mesures ne sont pas indépendantes : il n'est pas possible d'augmenter l'exactitude sans augmenter la qualité.

Dans le cas particulier des schémas Winograd, où il n'y a que deux réponses et où la collection est équilibrée par construction, un système répondant au hasard obtiendrait une exactitude de 50%. Ceci nous inspire une troisième mesure, que nous avons appelée *réussite* (Amsili & Seminck, 2017b) : c'est la mesure d'exactitude que donnerait un système qui répondrait au hasard pour tous les items correspondants à une non-réponse. La réussite s'interprète par comparaison avec le taux de 50% évoqué plus haut (niveau de la chance) : ce sont les points au-dessus de cette valeur qui nous renseignent sur la performance du système[4].

$$\textit{qualité} = \frac{h}{n-\theta} \qquad \text{exactitude} = \frac{h}{n} \qquad \text{réussite} = \frac{h+\frac{\theta}{2}}{n}$$

Nous présentons dans la table 1 les résultats de plusieurs systèmes, dont le nôtre, avec les différentes mesures évoquées à l'instant. Notons que si le système répond toujours, ce qui semble être le cas de celui de Trinh & Le (2018), $\theta = 0$ et toutes les mesures sont identiques. Les deux premières lignes de la table 1 présentent des résultats publiés par Emami *et al.* (2018) pour l'anglais. Le système AGQS, entièrement automatique, est présenté par ces derniers comme meilleur que le système de Sharma *et al.* (2015) sur la base de la mesure F1 qu'ils ont introduite (0,46 *vs.* 0,29). On peut critiquer cette conclusion, en observant que le système de Sharma *et al.* reste (légèrement) meilleur en terme de *réussite*. Il nous semble important de souligner de plus que le système MGQ de Emami *et al.*, cette fois-ci un système comprenant un traitement manuel, a certes de meilleurs résultats que AGQS quelle que soit la mesure, mais là encore il est contestable de poser qu'il dépasse celui de Sharma *et al.* (2015), du moins si on prend au sérieux la réussite telle que nous la proposons.

Nous donnons aussi dans la table 1 quelques-uns des résultats des travaux dont nous nous sommes directement inspirés (Trinh & Le, 2018) (toujours pour l'anglais). La seule mesure rapportée par les auteurs est l'exactitude, et nous en avons déduit que leur système est toujours capable de produire une réponse ($\theta = 0$). Cependant, peu de détails sont fournis (l'article est en cours d'évaluation). Si on

3. Situation dans laquelle la précision mesure à quel point le système est fiable lorsqu'il attribue la catégorie en question, et le rappel mesure à quel point le système est capable de retrouver tous les items appartenant à la catégorie. Il est bien établi que ces mesures sont indépendantes : pour augmenter la précision il faut réduire le nombre de *fausses alarmes* (fa), alors que pour augmenter le rappel il faut réduire le nombre de *manques* (m) (h étant ici encore le nombre de bonnes réponses).

$$\text{précision} = \frac{h}{h+fa} \qquad \text{rappel} = \frac{h}{h+m}$$

4. Une autre façon, peut-être plus intuitive, de mesurer la performance pourrait consister à attribuer $+1/n$ aux bonnes réponses, $-1/n$ aux mauvaises, et 0 aux non-réponses. Soit p cette mesure, qui va de -1 à 1 (elle vaut 0 si le système ne répond pas mieux que le hasard et 1 si le système répond toujours, et toujours correctement) ; on peut montrer qu'elle est définissable à partir de la réussite : $p = 2 \times \textit{réussite} - 1$.

	n	h	θ	exactitude *rappel** h/n	qualité *précision** $h/(n-\theta)$	*F1**	réussite $(h+\theta/2)/n$
Collection en anglais							
Emami *et al.* (2018) AGQS	273	106	83	38,83	55,79	45,79	54,03
Emami *et al.* (2018) MGQ	273	118	76	43,22	59,90	50,21	57,14
Sharma *et al.* (2015)	283	49	230	17,31	92,45	29,27	57,95
Trinh & Le (2018) Word-full	273	147	0	53,85	—*idem*—		
Trinh & Le (2018) 10 modèles	273	168	0	61,54	—*idem*—		
Trinh & Le (2018) 14 modèles	273	174	0	**63,74**	—*idem*—		
Collection en français							
LSTM (cet article)	214	65	88	30,37	51,59	38,24	50,93
C2V (cet article)	214	33	158	15,42	58,93	24,44	52,34
Amsili & Seminck (2017b)	180	72	49	40,00	54,96	46,30	53,61

*terminologie de Emami *et al.* (2018)

TABLE 1 – Comparaison des métriques (exactitude, qualité, réussite) pour les collections anglaise et française. Rappelons que si le système répond toujours ($\theta = 0$) les trois mesures sont identiques.

peut supposer que, disposant de vocabulaires beaucoup plus important que nous, les auteurs n'ont pas rencontré de non-réponses dues à un défaut de vocabulaire, il serait utile de savoir comment a été traité le problème des schémas multi-phrases qui sont à l'origine de beaucoup de non-réponses dans notre cas.

La première ligne (Word-full) correspond au modèle qui nous a inspiré le plus directement : un modèle de langue LSTM basé sur les mots (*vs.* un modèle de caractères) dont les prédictions se font sur vocabulaire total de 800K mots avec des représentations vectorielles de mots de l'ordre de 1024 dimensions. Dans leurs différentes expériences, les auteurs ont utilisé plusieurs corpus d'entraînement pour les modèles de langue : LM-1Billion, CommonCrawl, SQuAD et Gutemberg books, mais ils ne précisent pas sur quel corpus le modèle Word-Full a été entraîné. Le système à 10 modèles consiste en un assemblage de plusieurs modèles, mélangeant modèles de mots et modèles de caractères avec des variations au niveau des paramètres des modèles (corpus d'entraînement, optimisation, profondeurs des réseaux, *etc.*). Tous ces modèles de langue ont cependant en commun le fait que la couche de sortie des réseaux produit un résultat dans le même espace vectoriel (1024 dimensions) permettant ainsi d'assembler conjointement les modèles. Enfin, le système avec 14 modèles reprend l'assemblage des 10 modèles précédents en ajoutant 4 nouveaux modèles entraînés spécialement pour la tâche : les auteurs ont construit des données d'entraînement spécifiques en extrayant les documents du corpus CommonCrawl ayant le plus de mots en commun avec les tokens trouvées dans les schémas Winograd. On peut noter que, quelle que soit la métrique utilisée, les systèmes combinant 10 et 14 modèles obtiennent des performances nettement meilleures que les autres système évoqués. On peut cependant aussi remarquer qu'un modèle standard, pas spécifiquement adapté à la tâche, n'atteint pas, seul, des ordres de grandeur meilleurs que les autres système état de l'art.

Nous présentons enfin les résultats obtenus par nos deux modèles de langue, qui obtiennent un résultat à peine meilleur que le hasard. Notons tout de même que le modèle C2V obtient de meilleurs scores que LSTM, même si ce dernier a beaucoup moins de non-réponses. Nous pouvons conclure qu'un modèle de langue non adapté à la tâche ne permet pas d'avoir un gain comparable à celui que Trinh & Le (2018) obtiennent en combinant 10 voire 14 modèles.

Nous avons reproduit en dernière ligne les résultats rapportés par Amsili & Seminck (2017b) qui ont utilisé un algorithme très simple basé sur l'information mutuelle entre les réponses et les mots spécial et alternant. Il est intéressant de noter que nos modèles de langue ne semblent pas avoir de meilleur résultat que leur système, qui était développé non pas pour résoudre les schémas, mais pour vérifier que les schémas français étaient « Google-proofs » (non sensibles aux statistiques de co-occurrence). La table montre que cet algorithme obtient quand-même 3,6 points au dessus du hasard, ce qui peut suggérer que la collection française n'est pas entièrement insensible aux statistiques de co-occurence. [5] Il est possible par conséquent que les gains de nos modèles soient essentiellement dûs à des schémas non « Google-proofs » (on peut noter qu'aucune vérification systématique de la *Google-proofness* n'a été faite pour la collection anglaise). Si c'est le cas, alors la meilleure performance de C2V peut s'expliquer par une capacité à mieux capter les informations de co-occurrence en ciblant les réponses, alors que le modèle LSTM construit une représentation globale de la phrase.

4 Conclusion

Notre étude suggère que les modèles de langue simples génériques ne permettent pas de résoudre les schémas Winograd, et que les quelques points gagnés par rapport au hasard sont au moins autant dûs au fait que certains schémas ne sont pas entièrement Google-proofs qu'à une capacité à encoder les informations pertinentes. Il semble cependant possible d'obtenir des résultats meilleurs avec des modèles de langue, mais seulement en les utilisant de façon très sophistiquée (et très coûteuse), de manière à parvenir à encoder des connaissances du monde ou des connaissances spécifiques à la tâche. Par ailleurs, si la méthode semble fonctionner sans grandes difficultés pour l'anglais (l'article de Trinh & Le (2018) ne donne pas tous les détails), nous avons constaté que le transfert au français a nécessité un important travail en partie manuel qui nous semble réduire la généralité de la méthode.

Nous pensons que la résolution des schémas Winograd se trouve devant une alternative : il est peut-être possible de faire appel à des modèles de langue encore plus sophistiqués, et de réfléchir à la façon dont de tels modèles peuvent encoder le genre de connaissance nécessaire pour ces schémas, mais on peut se demander si cette approche qui reste intrinsèquement statistique n'est pas en train d'atteindre une asymptote. L'autre option serait de renoncer aux modèles de langue pour basculer vers des approches plus spécifiquement orientées vers l'encodage du raisonnement, ce pour quoi ces schémas ont été explicitement conçus.

Remerciements

Ce travail a reçu le soutien du *Labex* EFL (*Empirical Foundations of Linguistics*, ANR-10-LABX-0083). Nous remercions les relecteurs de TALN, ainsi que Chang Jiaqi, stagiaire du cursus de Linguistique Informatique à Paris Diderot.

5. Par exemple pour l'item *Un arbre est tombé sur le toit, il va falloir **le** déplacer* on peut s'attendre à des statistiques différentes de co-occurrence entre (*déplacer*, *toit*) et (*déplacer*, *arbre*).

Références

AMSILI P. & SEMINCK O. (2017a). A Google-proof collection of French Winograd schemas. In *Proceedings of the 2nd Workshop on Coreference Resolution Beyond OntoNotes (CORBON 2017), co-located with EACL 2017*, p. 24–29.

AMSILI P. & SEMINCK O. (2017b). Schémas Winograd en français : une étude statistique et comportementale. In *Conférence sur le Traitement Automatique du Langage Naturel*, volume 2, p. 28–35, Orléans : Association pour le Traitement Automatique des Langues.

COAVOUX M. (2017). *Discontinuous Constituency Parsing of Morphologically Rich Languages*. PhD thesis, Univ Paris Diderot, Sorbonne Paris Cité.

DUCHI J., HAZAN E. & SINGER Y. (2011). Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research*, **12**(Jul), 2121–2159.

EMAMI A., TRISCHLER A., SULEMAN K. & CHEUNG J. C. K. (2018). A generalized knowledge hunting framework for the Winograd schema challenge. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics : Student Research Workshop*, p. 25–31 : Association for Computational Linguistics.

HOCHREITER S. & SCHMIDHUBER J. (1997). Long short-term memory. *Neural computation*, **9**(8), 1735–1780.

LEVESQUE H., DAVIS E. & MORGENSTERN L. (2012). The Winograd schema challenge. In *Thirteenth International Conference on the Principles of Knowledge Representation and Reasoning*.

MELAMUD O., GOLDBERGER J. & DAGAN I. (2016). context2vec : Learning generic context embedding with bidirectional LSTM. In *Proceedings of The 20th SIGNLL Conference on Computational Natural Language Learning*, p. 51–61.

SHARMA A., VO N. H., ADITYA S. & BARAL C. (2015). Towards addressing the Winograd schema challenge-building and using a semantic parser and a knowledge hunting module. In *Proceedings of Twenty-Fourth International Joint Conference on Artificial Intelligence. AAAI*.

TRINH T. H. & LE Q. V. (2018). A Simple Method for Commonsense Reasoning. *ArXiv e-prints*.