Interprétation de prédictions de modèles pour l'inférence textuelle en perturbant significativement les entrées

Marjorie Armando

Laboratoire d'Informatique et des Systèmes - LIS

Encadrant

Benoit Favre

Résumé

La reconnaissance de l'inférence textuelle (Recognizing Textual Entailment : RTE) est un domaine assez récent en traitement automatique du langage (TAL). Le but de la RTE est de savoir automatiquement si une phrase, appelée l'hypothèse, est déduite d'une autre phrase, appelée la prémisse. Pour résoudre cela, on utilise des systèmes issus de l'apprentissage automatique.
Le problème qui se pose avec l'utilisation de ces systèmes est que le modèle nous fournit uniquement les probabilités de chaque label. Ainsi, aucune information supplémentaire n'est donnée : comment savoir, de manière humainement compréhensible, pourquoi le modèle a associé un label particulier à une paire de phrases donnée ?
Pouvoir répondre à cette question permettrait de rendre un modèle utilisable dans des domaines où les décisions doivent être mûrement réfléchies telle que la médecine, car malgré l'expansion des réseaux de neurones, ceux-ci restent des boîtes noires et il est donc difficile de leur faire confiance sans avoir d'explications en retour.
Dans ce travail, nous proposons une méthode respectant les règles d'une "bonne" explication pour rendre un modèle interprétable dans le cadre de la RTE avec l'utilisation du corpus SNLI [1]. Nous allons la comparer avec la méthode Local Interpretable Model-agnostic Explanations (LIME) [2] qui permet d'expliquer les prédictions de n'importe quel classifieur en apprenant localement un modèle interprétable dans le voisinage de l'entrée.

Mots-clés : inférence textuelle, apprentissage automatique,
traitement automatique des langues naturelles, interprétabilité, LSTM

1 Introduction

1.1 Contexte de l'étude

Le traitement automatique du langage naturel ou de la langue naturelle (TALN) ou des langues (TAL) est un domaine pluridisciplinaire, qui fait collaborer l'intelligence artificielle, l'informatique théorique, la logique, la linguistique ou encore les statistiques en vue de modéliser et de reproduire à l'aide de machines, la capacité humaine à produire et à comprendre les énoncés linguistiques dans des buts de communication.

Dans le domaine du TAL, nous retrouvons plusieurs niveaux d'analyses linguistiques pour représenter au mieux les langues naturelles par les machines. On peut par exemple citer l'analyse lexicale qui permet d'identifier quels sont et où sont les mots, ou encore l'analyse sémantique qui permet de comprendre le sens des mots.

La RTE permet d'apporter des méthodes pour l'analyse lexicale et sémantique. Ceci peut permettre le développement de diverses applications telles que la recherche d'information ou encore le système de question/réponse.

Le but de la RTE est de savoir automatiquement si, à partir de deux phrases, on peut en déduire la deuxième de la première. La première phrase est appelée la prémisse, et la seconde l'hypothèse. Trois étiquettes permettent d'illustrer la relation entre la prémisse et l'hypothèse :

- **Contradiction** : l'hypothèse contredit la prémisse.
  Exemple :
  Prémisse : "Le chat est entièrement blanc."
  Hypothèse : "Le chat est entièrement noir."
- **Neutre** : l'hypothèse est possible dans le contexte de la prémisse.
  Exemple :
  Prémisse : "Le chat dort sur la banquette."
  Hypothèse : "Le chat aime le chocolat."
- **Inférence** : l'hypothèse est déduite de la prémisse.
  Exemple :
  Prémisse : "Le chat aimerait manger la sourie."
  Hypothèse : "Le chat a faim."

Pour résoudre cela, on utilise des systèmes issus de l'apprentissage automatique. Nous utilisons les réseaux de neurones récurrents (Recurrent Neural Network : RNN) dans ce projet.

1.2 Problématique

Le problème qui se pose avec l'utilisation de système issue de l'apprentissage automatique est que le modèle nous fournit uniquement les probabilités de chaque label. Comment savoir, de manière humainement compréhensible, pourquoi le modèle a associé un label particulier à une paire de phrases donnée ?

Pouvoir répondre à cette question permettrait de rendre un modèle utilisable dans des domaines où les décisions doivent être mûrement réfléchies, telle que la médecine. Si un modèle propose de donner un certain traitement à un patient, il faut que le modèle puisse donner de bonnes explications pour que le docteur l'approuve, car les conséquences pourraient être catastrophique.

Ainsi, un modèle doit être digne de confiance. Cette confiance donnée par les humains à un système dépend des explications données. Nous verrons dans la partie "Interprétabilité" quels sont les critères d'une "bonne" explication. De plus, si un système est jugé digne de confiance, il sera davantage utilisé : en effet, il a été observé, par exemple, que le fait de fournir des explications augmente l'acceptation des recommandations de films [2].

Pouvoir donner une interprétation du système peut donc permettre aux utilisateurs d'adopter un modèle. Mais cela peut également aider le développeur lors du choix d'un modèle parmi ceux qu'il a implémenté, car le taux de réussite n'est pas le seul critère à prendre en compte : un modèle peut fournir le label attendu en se basant sur de mauvaises features.

Dans ce rapport, nous allons tout d'abord énumérer quelques méthodes existantes pour l'interprétation de prédictions d'un modèle. Nous allons ensuite spécifier les modèles que nous avons implémenté pour la RTE ainsi que les corpus utilisé, puis nous allons définir ce que nous devons attendre d'une "bonne" explication. Puis, nous allons expliciter notre technique pour l'interprétation d'une prédiction d'un modèle, et nous allons la comparer avec LIME. Enfin, nous allons conclure avec les différentes perspectives.

2 Etat de l'art

2.1 Dans le domaine de la RTE

Avant la publication du corpus SNLI [1], la recherche sur l'apprentissage automatique dans ce domaine été limitée par le manque de corpus assez grand. Depuis, les plus grands taux de réussite ont atteint 89,3% [3].

2.2 Dans l'interprétabilité

Dans la pratique, il y a souvent un compromis entre le taux de réussite et l'interprétabilité du modèle. Certains modèles tels que les arbres de décisions ou encore les modèle linéaires sont facilement interprétables, et sont donc parfois utilisés à la place de modèles complexes tels que les réseaux de neurones profonds, même si ceux-ci peuvent donner de meilleurs résultats. Cependant, au cours de ces dernières années, de grandes avancées ont été effectuées pour tenter d'interpréter les modèles utilisés jusqu'alors comme des boîtes noires.

Nous étudions dans ce rapport la méthode LIME qui permet d'expliquer les prédictions de n'importe quel classifieur ou regresseur. L'objectif global de LIME est d'identifier un modèle interprétable parmi le voisinage de l'entrée x.

Tout d'abord, on distingue les features utilisées aux représentations interprétables des features. Par exemple, les features sont les embeddings des mots et la représentation interprétable de ces features est un vecteur binaire qui indique la présence ou l'absence des mots.

LIME définit une explication par un modèle $g \in G$, où G est la classe des modèles interprétables tels que les modèles linéaires ou les arbres de décisions. Vu que les modèles interprétables n'ont pas tous la même difficulté à être interprété, LIME définit $\Omega(g)$ qui est une mesure de la complexité d'interpréter g. En prennant l'example des arbres de décisions, $\Omega(g)$ est la profondeur.
On dénote par $f : \mathbb{R}^d \to \mathbb{R}$ le modèle utilisé comme une boîte noire. $f(x)$ est la probabilité que l'entrée x appartienne à une certaine classe.
LIME va alors se baser sur la représentation interprétable des données en retirant un mot au hasard. On dénote cette nouvelle entrée par z. On définit la localité de x avec $\pi_x(z)$ qui est une mesure de proximité entre z et x. C'est un noyau se basant sur la similarité cosinus.
Enfin, on définit $L(f, g, \pi_x)$ qui est une mesure pour savoir à combien g est infidèle à f dans la localité défini par π_x. Pour préserver à la fois l'interprétabilité et la fidèlité locale, LIME minimise $L(f, g, \pi_x)$ avec $\Omega(g)$ assez petit pour être interprétable par les humains. L'explication de LIME est donc la suivante :

$$\varepsilon(x) = \underset{g \in G}{argmin} L(f, g, \pi_x) + \Omega(g) \tag{1}$$

Cette formule peut être utilisée par différents modèles $g \in G$, fonctions de fidèlité $L(f, g, \pi_x)$, et mesure de complexité $\Omega(g)$.

LIME peut alors donner les K mots les plus importants de l'entrée x pour tout label.
La figure ci-dessus est un exemple illustant le principe de LIME :

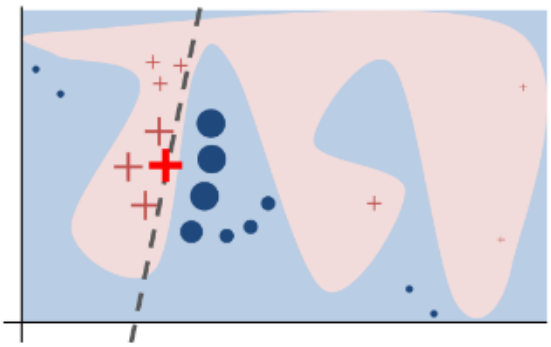

FIGURE 1 – Exemple présentant l'intuition de LIME.

La décision de la boîte noire f, inconnu par LIME, est représentée par le fond bleu et rose. La croix rouge en gras est l'entrée x que l'on veut expliquer. LIME crée des entrées modifiées, utilise f pour avoir la probabilité de ces entrées pour le label y, et les pondère par leur proximité par rapport à x (les poids sont représentés par la taille). La droite pointillée est l'explication apprise qui

est localement fidèle.

2.3 Dans l'interprétabilité de la RTE

Ce n'est que très recemment que /*PARLER ARTICLE LREC*/

3 Implémentation de systèmes

3.1 LSTM et Bi-LSTM

Dans ce travail, nous utilisons les réseaux de neurones récurrents, et plus particulièrement les LSTMs (Long Short-Term Memory : LSTM) et les Bi-LSTMs (Bidirectionnal Long Short-Term Memory : Bi-LSTM).
Ils permettent de relier des informations vues antérieurement à la tâche courante. On leur donne des informations à chaque instant t, créant ainsi une cellule.

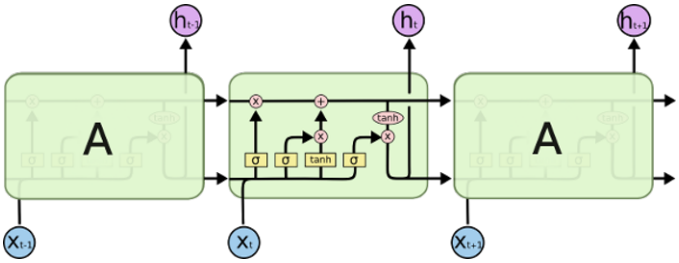

FIGURE 2 – Schéma d'un LSTM contenant trois cellules.

Ce sont des RNNs particuliers capable d'apprendre une dépendance très éloignée à notre tâche courante : ils ont la capacité de supprimer ou d'ajouter des informations à l'état de la cellule, régulé par des portes qui décident s'il faut laisser passer de l'information.
Ces réseaux de neurones sont utilisés ici en tant qu'encodeur : on fournit en entrée, à chaque instant t, l'embedding du $t^{\text{ième}}$ mot de la phrase traitée, à savoir la prémisse ou l'hypothèse. Le réseau calcule alors à chaque instant t un état caché h_t comme suit :

$$i_t = \sigma(W_i x_t + U_i h_{t-1} + b_i) \tag{2}$$
$$f_t = \sigma(W_f x_t + U_f h_{t-1} + b_f) \tag{3}$$
$$u_t = \tanh(W_u x_t + U_u h_{t-1} + b_u) \tag{4}$$
$$o_t = \sigma(W_o x_t + U_o h_{t-1} + b_o) \tag{5}$$
$$c_t = f_t \odot c_{t-1} + i_t \odot u_t \tag{6}$$
$$h_t = o_t \odot \tanh(c_t) \tag{7}$$

où σ est la fonction sigmoid, $\odot$ est le produit de Hadamard entre deux vecteurs, W_i, W_f, W_u, $W_o \in \mathbb{R}^{D \times D_e}$, U_i, U_f, U_u, $U_o \in \mathbb{R}^{D \times D}$ et b_i, b_f, b_u, $b_o \in \mathbb{R}^D$ sont des paramètres mis-à-jour par le réseau. D est la dimension des états cachés et D_e est la dimension des embeddings.

3.2 Systèmes implémentés

Nous avons implémenté 3 systèmes différents avec la librairie DyNet [4] en C++ :

Le premier système passe la prémisse et l'hypothèse au LSTM pour avoir une représentation pour chacune de ces deux phrases. On les concatène pour les envoyer ensuite à une couche de décision :

$$y = softmax(W \times [LSTM(prémisse)\ ; LSTM(hypothèse)] + b) \tag{8}$$

où y est un vecteur contenant les probabilités de chaque label, W est la matrice de poids, $LSTM(prémisse)$ et $LSTM(hypothèse)$ sont respectivement la représentation de la prémisse et de l'hypothèse, b est le biais, et [;] dénote la concaténation.

Le deuxième système effectue le même mécanisme que le premier système pour avoir une représentation de la prémisse et de l'hypothèse. On compare les deux représentations pour envoyer la comparaison à une couche de décision :

$$y = softmax(W \times (LSTM(prémisse) \times LSTM(hypothèse)^T) + b) \tag{9}$$

où T dénote la transposée.

Le troisième système est inspiré de la méthode KIM [5].
On représente les mots de la prémisse et de l'hypothèse en les passant dans un Bi-LSTM : il utilise un LSTM forward pour lire la phrase de gauche à droite, puis un LSTM backward pour lire la phrase dans l'autre sens. A chaque mot lu, un état caché est généré par les deux LSTMs. Ces deux états cachés sont alors concaténés pour obtenir une représentation du mot :
$h_t = [\overrightarrow{h_t}; \overleftarrow{h_t}]$, où $\overrightarrow{h_t}$ est l'état caché généré par le LSTM forward à l'instant t, $\overleftarrow{h_t}$ est celui généré par le LSTM backward à l'instant t, et h_t est la représentation du mot t.
On dénote par p^s (respectivement h^s) le vecteur de représentation des mots de la prémisse (respectivement de l'hypothèse).
Nous construisons ensuite une matrice d'alignement comme suit :

$$e_{ij} = (p_i^s)^T h_j^s \tag{10}$$

où p_i^s est la représentation du i^{ème} mot de la prémisse, h_j^s est celle du j^{ième} mot de l'hypothèse.
Avec cette matrice, nous pouvons alors construire les vecteurs de contexte p^c et h^c suivants pour la prémisse et l'hypothèse :

$$\alpha_{ij} = \frac{exp(e_{ij})}{\sum_{k=1}^{N} exp(e_{ik})}, p_i^c = \sum_{j=1}^{N} \alpha_{ij} h_j^s \tag{11}$$

$$\beta_{ij} = \frac{exp(e_{ij})}{\sum_{k=1}^{M} exp(e_{kj})}, h_j^c = \sum_{i=1}^{M} \beta_{ij} p_i^s \tag{12}$$

où M est la longueur de la prémisse, N est la longueur de l'hypothèse, $\alpha \in \mathbb{R}^{M \times N}$ est un $softmax(e)$ sur la prémisse, et $\beta \in \mathbb{R}^{M \times N}$ est un $softmax(e)$ sur l'hypothèse. Ceci permet à la prémisse de voir le contexte de l'hypothèse et vice-versa.
Avec ces nouvelles représentations pour les mots, on effectue du mean-pooling :

$$pool_p = \frac{\sum_{i=1}^{N} p_i^c}{N}, pool_h = \frac{\sum_{i=1}^{M} h_i^c}{M} \tag{13}$$

puis on effectue une concaténation du mean-pooling de la prémisse et de l'hypothèse pour l'envoyer à une couche de décision :

$$y = softmax(W \times [pool_p; pool_h] + b) \tag{14}$$

Les performances de ces systèmes et autres détails techniques sont décrits dans la partie "résultats expérimentaux" en annexe.
Les schémas de ces systèmes figurent dans la partie "" en annexe.

3.3 Données externes utilisées

Pour la RTE, nous utilisons les corpus SNLI composés d'un fichier d'entraînement, de validation et de test. Chaque fichier est en format json et en texte brut. Nous avons donc pu facilement les tokeniser.
Ces corpus ont été réalisé par 5 annotateurs à l'aide d'une image accompagnée d'un texte bref -la prémisse- présentant la dite image. Les annotateurs devaient alors écrire une phrase étant neutre par rapport à la prémisse, une autre étant en contradiction et une autre phrase qui pouvait être déduite de la prémisse : ils devaient donner une hypothèse et un label. Pour que le corpus ne soit pas trop subjectif, les annotateurs ont eu accès à quelques paires prémisse/hypothèse sans label. Chacun d'entre eux a donné un label, celui ayant eu le plus de voix a été décidé comme le label de la paire observée. Ainsi, certaines paires n'ont pas de label car les annotateurs n'ont pas trouvé de consensus : nous ne prennons pas en compte ce genre d'entrée.

La table ci-dessus est un échantillon du corpus SNLI :
/*faire belle table avec exemple ayant plusieurs labels*/
Concernant les embeddings des mots, nous utilisons les embeddings pré-entraînés de GloVe. Les mots ne se trouvant pas dans le vocabulaire de GloVe ont des embeddings initialisés au hasard.

4 Définition d'interprétabilité

Il n'y a malheureusement pas de consensus concernant la définition d' "interprétabilité". Miller définit cela comme étant le degré auquel un humain peut comprendre la cause d'une décision[6]. Un système a donc une meilleure interprétabilité qu'un autre si ses explications sont plus faciles à comprendre par un humain. Par ailleurs, on peut interpréter de manière globale en expliquant le comportement général du modèle, ou de manière locale en expliquant pourquoi le modèle a choisi tel label. Dans ce travail, nous nous placons dans le cas de l'interprétation locale : le but est d'expliquer pourquoi le modèle a choisi un label y pour une entrée x.

4.1 Qu'est-ce-qu'une explication ?

La définition donnée par Miller est assez simple : une explication est une réponse à une question commençant par "pourquoi". Une question commençant par "comment" peut être retournée en une question commençant par "pourquoi". Le terme "explication" désigne le processus social et cognitif d'expliquer, mais c'est également le produit de ces processus.

4.2 Qu'est-ce-qu'une "bonne" explication ?

La définition d'une bonne explication ne doit pas se baser sur l'intuition de l'auteur, mais plutôt sur des faits. Miller résume ce qu'est une bonne explication[7], basée sur ce que les humains attendent d'une explication. Grâce à cela, nous allons énoncer les types d'explications adéquats à notre projet. Cependant, il ne faut pas oublier que les humains ont tendance à rejeter toutes explications allant à l'encontre de leur croyance.

4.2.1 Explication contrastée

C'est une explication qui doit être comparée. Les utilisateurs se demandent généralement pourquoi cette prédiction a été faite et pas une autre, via la question "quelle aurait été la prédiction si cette entrée avait été changé par une autre ?".
Un docteur se demandant "pourquoi ce traitement ne marche pas sur ce patient ?" voudrait comparer les données de ce patient à un autre patient ayant des caractéristiques similaires mais pour qui le traitement marche.
La meilleure explication pour ce type d'explication est celle qui met en évidence les différences entre l'entrée traitée et l'entrée de comparaison.
L'entrée de comparaison peut être artificielle.

4.2.2 Explication sélective

C'est une explication qui doit être courte. Généralement, on peut expliquer un phénomène par plusieurs facteurs. Il faut en donner peu, à savoir deux ou trois raisons, même si les explications peuvent être plus complexes que cela.

4.2.3 Explication sociale

Comme nous l'avons expliqué ci-dessus, une explication est un processus social, c'est-à-dire qu'il faut prendre en compte les connaissances de la personne à qui l'on veut donner une explication. Dans notre projet, nous partons du principe qu'une explication doit être comprise par tout le monde, que ce soit par un expert du domaine de l'apprentissage automatique ou bien par quelqu'un qui n'en a jamais entendu parler.

5 Techniques pour l'interprétabilité

Dans cette section nous allons décrire notre technique pour interpreter une prédiction d'une entrée. Nous voulons donner des explications pour chaque label en donnant les trois mots les plus importants dans la prémisse et les trois mots les plus importants dans l'hypothèse. On veut donc calculer l'importance

de chaque mot. L'importance d'un mot correspond à l'importance de sa contribution pour le label y.
Notre intuition est la suivante : On veut donner une explication pour le label y et l'entrée x composée de la prémisse et de l'hypothèse. On retire alors un mot et on demande à notre modèle de nous donner les probabilités de chaque label avec cette nouvelle entrée. Si la probabilité du label y a baissé, alors le mot était important : cela veut dire que le mot avait contribué au label y. A l'inverse, si elle a augmenté, le mot n'avait donc pas contribué au label y. De plus, si la probabilité des autres labels a augmenté, alors le mot a d'autant plus d'importance : en le retirant, l'entrée x a basculé vers un autre label.

L'importance d'un mot, que nous notons IF pour Impact Factor, est calculé comme suit :

$$IF^{y_j}(m_i) = \sum_{j=1}^{|Y|} \log(p(y_j|x\backslash m_i)) - \log(p(y_j|x)) \tag{15}$$

$$IF^{y_{ref}}(m_i) = -\log(p(y_{ref}|x\backslash m_i)) + \log(p(y_{ref}|x)) \tag{16}$$

$$IF^{total}(m_i) = IF^{y_{ref}}(m_i) + IF^{y_j}(m_i) \tag{17}$$

où y_{ref} est le label de référence (le label que l'on veut "expliquer"), x est l'entrée, m_i est le mot que l'on retire de l'entrée x, y_j est un label différent de y_{ref}, et $|Y|$ est le nombre de label. Les probabilités sont exprimées en log-probabilités pour des soucis de précision et d'optimisation du code. On parle donc plutôt de score.
Cette formule suit notre intuition de base et également celle de Robnik-Sikonja et Kononenko [8]. Nous avons rajouté la prise en compte de l'impact sur les scores des autres labels quand on retire le mot m_i.

Un problème persiste tout de même : que se passe-t-il vraiment lorsque l'on retire un mot ? Nous ne pouvons pas simplement le supprimer de l'entrée, car la phrase n'aurait plus de sens et le modèle n'est peut-être pas entraîné à faire face aux bruits. On ne peut pas vraiment le remplacer par un mot générique (le mot "UNK" pour "mot inconnu", par exemple) car un modèle remplacant ce mot par un embedding à 0 et un modèle remplacant ce mot par un embedding particulier auront des réponses très différentes. Or on voudrait que notre méthode soit réalisable par n'importe quel modèle. La solution est donc de remplacer le mot que l'on veut retirer par un autre :

$$p(y|x\backslash m_i) = \sum_{s=1}^{r_i} p(y|x \leftarrow m_i = m_s) \tag{18}$$

où r_i est le nombre de mot que l'on peut mettre à la place du mot m_i et m_s est le mot que l'on met à la place de m_i.
Les mots pouvant remplacer m_i sont situés dans un fichier externe créé par nous-même. On a fixé à 5 le nombre de mots maximum pouvant remplacer m_i. Plus il y a de mots de remplacements, plus on a une meilleure estimation du IF du mot m_i, mais dans ce cas la complexité en temps augmente. Il faut donc trouver

un compromis entre l'estimation du IF et le temps que met le programme pour donner une explication.

Cette méthode permet donc d'avoir une explication contrastée, puisque l'on compare notre entrée de base avec des entrées créées artificiellement en remplacant un mot par un autre. De plus, elle est également sélective puisque l'on sélectionne les K mots, respectivement dans la prémisse puis dans l'hypothèse, ayant le plus contribué au label y_{ref}. Enfin, le programme surligne les mots les plus importants pour chaque label, ce qui permet d'avoir une visualisation et donc d'être compris par n'importe qui.

/* METTRE UN SCREEN D'UNE EXPLICATION */

Références

[1] Samuel R. Bowman, Gabor Angeli, Christopher Potts, and Christopher D. Manning. A large annotated corpus for learning natural language inference. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Association for Computational Linguistics, 2015.

[2] Marco T. Ribeiro, Sameer Singh, and Carlos Guestrin. 'why should i trust you ?' : Explaining the predictions of any classifier. In *International Conference on Knowledge Discovery and Data Mining (KDD)*, 2016.

[3] Reza Ghaeini, Sadid A. Hasan, Vivek Datla, Joey Liu, Kathy Lee, Ashequl Qadir, Yuan Ling, Aaditya Prakash, Xiaoli Z. Fern, and Oladimeji Farri. Dr-bilstm : Dependent reading bidirectional lstm for natural language inference. In *16th Annual Conference of the North American Chapter of the Association for Computational Linguistics : Human Language Technologies (NAACL HLT)*, 2018.

[4] Graham Neubig, Chris Dyer, Yoav Goldberg, Austin Matthews, Waleed Ammar, Antonios Anastasopoulos, Miguel Ballesteros, David Chiang, Daniel Clothiaux, Trevor Cohn, Kevin Duh, Manaal Faruqui, Cynthia Gan, Dan Garrette, Yangfeng Ji, Lingpeng Kong, Adhiguna Kuncoro, Gaurav Kumar, Chaitanya Malaviya, Paul Michel, Yusuke Oda, Matthew Richardson, Naomi Saphra, Swabha Swayamdipta, and Pengcheng Yin. Dynet : The dynamic neural network toolkit. *arXiv preprint arXiv :1701.03980*, 2017.

[5] Qian Chen, Xiaodan Zhu, Zhen-Hua Ling, Diana Inkpen, and Si Wei. Natural language inference with external knowledge. *arXiv preprint arXiv :1711.04289*, 2017.

[6] Tim Miller. Explanation in artificial intelligence : Insights from the social sciences. *arXiv preprint arXiv :1706.07269*, 2017.

[7] Christoph Molnar. *Interpretable Machine Learning*. Creative Commons Attribution-NonCommercial-ShareAlike 4.0 International License, 2018.

[8] Marko Robnik-Sikonja and Igor Kononenko. Explaining classifications for individual instances. In *IEEE Transactions on Knowledge and Data Engineering*, 2008.

A Résulats expérimentaux

Pour la représentation des mots, nous utilisons des words embeddings pré-entrainés de dimension 100 via GloVe.6B.100d. Pour les mots inconnus, c'est-à-dire les mots qui n'ont pas d'embedding dans GloVe, nous utilisons des embeddings initialisés au hasard. Tous les embeddings sont mis-à-jour par le réseau.
La taille des batches est de 16 et le dropout est à 0,3.
Concernant les RNNs utilisés, il n'y a qu'une seule couche et la dimension des états cachés est de 100.

Système	Taux de réussite Contradiction	Taux de réussite Inférence	Taux de réussite Neutre	Taux de réussite Dev	Tau
1	67,33%	73,90%	68,38%	69,89%	test
2	78,55%	87,98%	74,99%	80,57%	test
3	68,73%	74,92%	69,55%	71,09%	test