TER: Modèles neuronaux pour le traitement des langues

Raisonnement automatique question/réponse et notation automatique

Thierry Loesch Bryce Tichit

M1 Informatique
Université Paris-Sud

Avril 2017

université PARIS-SACLAY

Sommaire

Introduction

Les **réseaux de neurones** sont un outil formidable lorsqu'il s'agit du *traitement automatique de la langue*.

Introduction

Les **réseaux de neurones** sont un outil formidable lorsqu'il s'agit du *traitement automatique de la langue*.

Afin de répondre aux exigences actuelles en matière d'apprentissage sur texte, il est nécessaire de synthétiser les données et développer une réelle méthode de compréhension.

Introduction

Les **réseaux de neurones** sont un outil formidable lorsqu'il s'agit du *traitement automatique de la langue*.

Afin de répondre aux exigences actuelles en matière d'apprentissage sur texte, il est nécessaire de synthétiser les données et développer une réelle méthode de compréhension.

Dans ce projet nous mettons en oeuvre une méthodologie et un modèle afin d'inférer dans un premier temps une réponse à une question portée sur un texte.

Introduction

Les **réseaux de neurones** sont un outil formidable lorsqu'il s'agit du *traitement automatique de la langue*.

Afin de répondre aux exigences actuelles en matière d'apprentissage sur texte, il est nécessaire de synthétiser les données et développer une réelle méthode de compréhension.

Dans ce projet nous mettons en oeuvre une méthodologie et un modèle afin d'inférer dans un premier temps une réponse à une question portée sur un texte.

Dans un deuxième temps nous essayerons de transposer la méthode précedente à un autre problème plutôt similaire : la *notation automatique de réponses d'étudiants*. Le but sera là d'inférer une note en fonction des réponses.

Projet Babi Tasks

Le projet Babi Tasks : Consiste en un projet de 20 tâches avec pour objectif de raisonner sur des phrases écrites, modéliser un énoncé et inférer un mot

Projet Babi Tasks

Le projet Babi Tasks : Consiste en un projet de 20 tâches avec pour objectif de raisonner sur des phrases écrites, modéliser un énoncé et inférer un mot

Les tâches sont diverses, allant du raisonnement sur une question à partir de un, deux ou trois faits justificatifs jusqu'au raisonnement sur des faits temporels.

Description des données

Exemples de tâches et données associées :

Task 1: Single Supporting Fact

Mary went to the bathroom.
John moved to the hallway.
Mary travelled to the office.
Where is Mary? A:office

Task 2: Two Supporting Facts

John is in the playground.
John picked up the football.
Bob went to the kitchen.
Where is the football? A:playground

Task 3: Three Supporting Facts

John picked up the apple.
John went to the office.
John went to the kitchen.
John dropped the apple.
Where was the apple before the kitchen? A:office

Task 4: Two Argument Relations

The office is north of the bedroom.
The bedroom is north of the bathroom.
The kitchen is west of the garden.
What is north of the bedroom? A: office
What is the bedroom north of? A: bathroom

Task 5: Three Argument Relations

Mary gave the cake to Fred.
Fred gave the cake to Bill.
Jeff was given the milk by Bill.
Who gave the cake to Fred? A: Mary
Who did Fred give the cake to? A: Bill

Task 6: Yes/No Questions

John moved to the playground.
Daniel went to the bathroom.
John went back to the hallway.
Is John in the playground? A:no
Is Daniel in the bathroom? A:yes

Description des données

Vocabulaire maîtrisé : Le vocabulaire est maîtrisé et de taille assez restreinte

Description des données

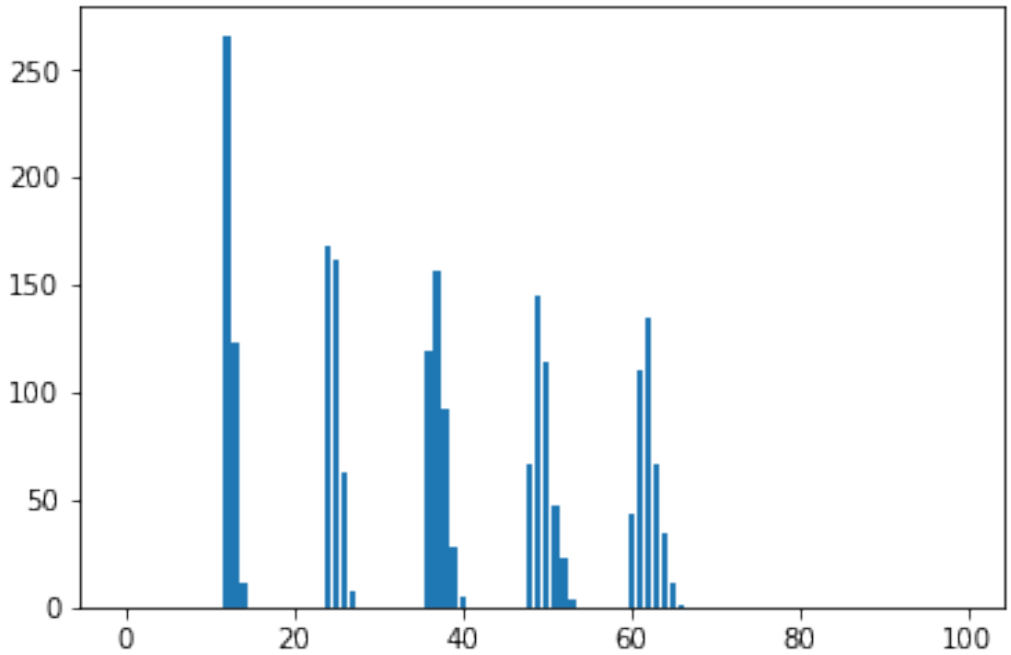
Vocabulaire maîtrisé : Le vocabulaire est maîtrisé et de taille assez restreinte

Généré par un algorithme *Torch*.

Description des données

Vocabulaire maîtrisé : Le vocabulaire est maîtrisé et de taille assez restreinte

Généré par un algorithme *Torch*.



Méthodologie

Pour parvenir à notre objectif nous utiliserons entre autres,

- Keras, une surcouche pour Theano et Tensorflow utilisé pour notre modèle
- Les embeddings
- Les réseaux de neurones récurrents

Word Embeddings

Méthode d'apprentissage automatique issue du deep learning reposant sur l'apprentissage d'une représentation de mot.

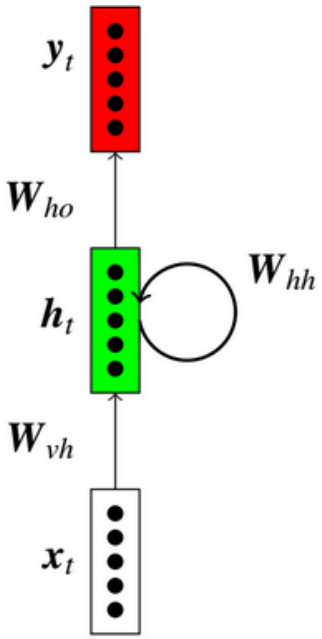
Cette méthode à permis de révolutionner l'apprentissage automatique de la langue.

Permet de se faire une représentation d'une phrase sous forme de vecteur. Celui-ci est beaucoup plus petit que s'il fallait stocker la phrase entière $\Rightarrow$ permet de condenser les particularités d'un texte.

En conséquence, il en suit un effort d'apprentissage réduit.

Réseaux récurrents

Pour *raisonner* sur un texte nous avons un outil indispensable : les **réseaux récurrents**.



Pour chaque instant t :

- maintient une représentation interne de l'historique h_t
- Mise à jour du réseau à partir d'une observation x_t et de l'état de l'historique précédent h_{t-1}
- La prédiction y_t dépend de l'historique h_t
- L'entrée du réseau vient des embeddings

Réseaux récurrents

Différents types de réseaux récurrents,

- **G**ated **R**ecurrent **U**nit
- **L**ong **S**hort **T**erm **Memory** Networks
- Memory Networks
- ...

Dans ce projet nous utiliserons les réseaux LSTM, toutefois il a été montré dans l'article[1] que les Memory Networks étaient très clairement meilleurs pour ce projet. Ceux-ci permettent de voir plus clairement à travers le *bruit* des données.

Vectorisation

Nous commençons par coder nos données avec des vecteurs et en mettant le tout dans des matrices : c'est la **vectorisation**.

Exemple :

```
[u'Where', u'is', u'Mary', u'?']
=>
[9, 15, 7, 4]
```

Vectorisation

Nous commençons par coder nos données avec des vecteurs et en mettant le tout dans des matrices : c'est la **vectorisation**.

Exemple :

```
[[ 0  9 15  7  4]
 [ 0  9 15  5  4]
 [ 0  9 15  5  4]
 [ 2  9 15  5  4]
 [ 3  9 15  8  4]
 [ 0  9 15  8  4]
 [ 0  9 15  8  4]
 [ 0  9 15  8  4]
 [ 2  9 15  6  4]
 [ 3  9 15  5  4]]
```

On applique la même opération sur l'ensemble de nos données.

Keras

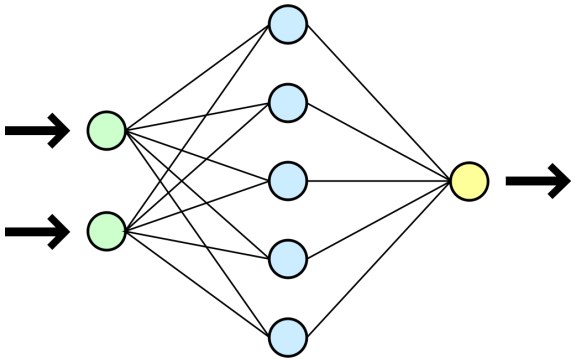
Passons au **modèle**,

Il s'agit du système qui va modéliser notre énoncé, nous utilisons *Keras* pour ce faire. Il s'agit d'une surcouche pour *Theano* et *Tensorflow* qui permet de créer facilement des réseaux de neurones à souhait.

De nombreux outils très puissants de **Machine Learning** sont implémentés dans la librairie *Keras* par défaut.

Keras

Exemple de réseau de neurones écrit en Keras :



```
model = Sequential()
model.add(Dense(5,input_shape=(2,)))
model.add(Dense(1))
model.add(Activation("tanh"))
```

Il est par la suite très simple d'entraîner notre réseau grâce à la fonction **Model.Fit**

Modèle

La particularité du projet **Babi Tasks** : devoir traiter deux données à la fois.

- Les histoires
- Les questions portant sur les histoires
- Et bien évidemment la donnée Y : la réponse à la question

Comment faire pour implémenter cela en Keras sachant qu'il faudra traiter chaque donnée différemment ?

Modèle

La particularité du projet **Babi Tasks** : devoir traiter deux données à la fois.

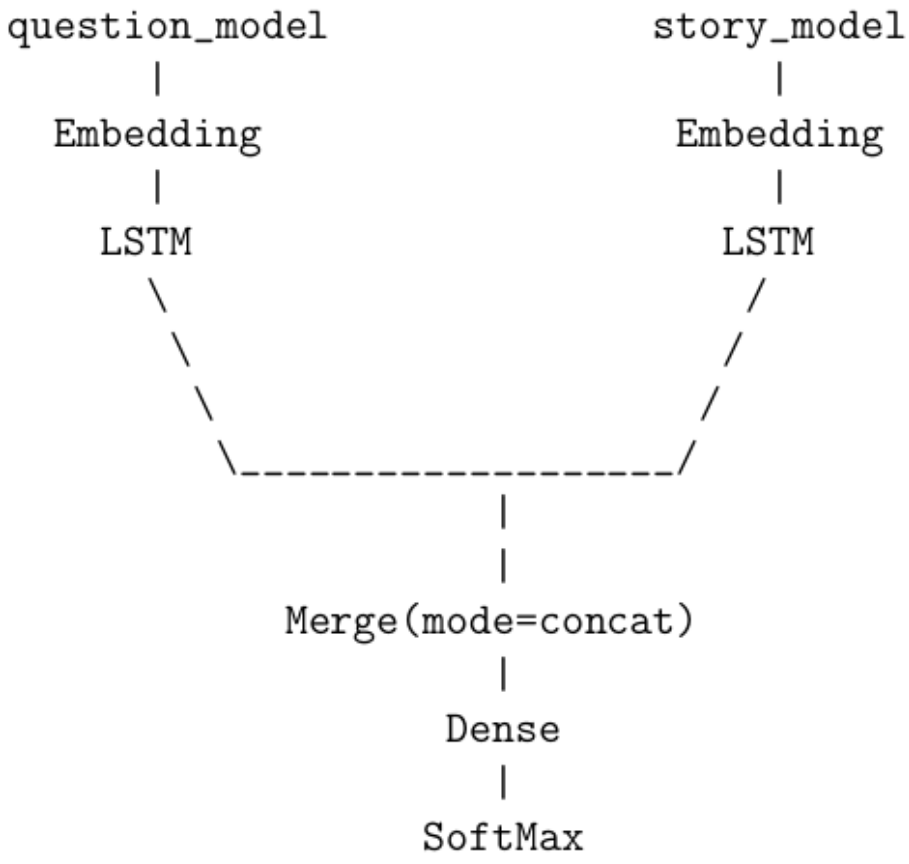
- Les histoires
- Les questions portant sur les histoires
- Et bien évidemment la donnée Y : la réponse à la question

Comment faire pour implémenter cela en Keras sachant qu'il faudra traiter chaque donnée différemment ?

Grâce à la couche **Merge** de Keras !

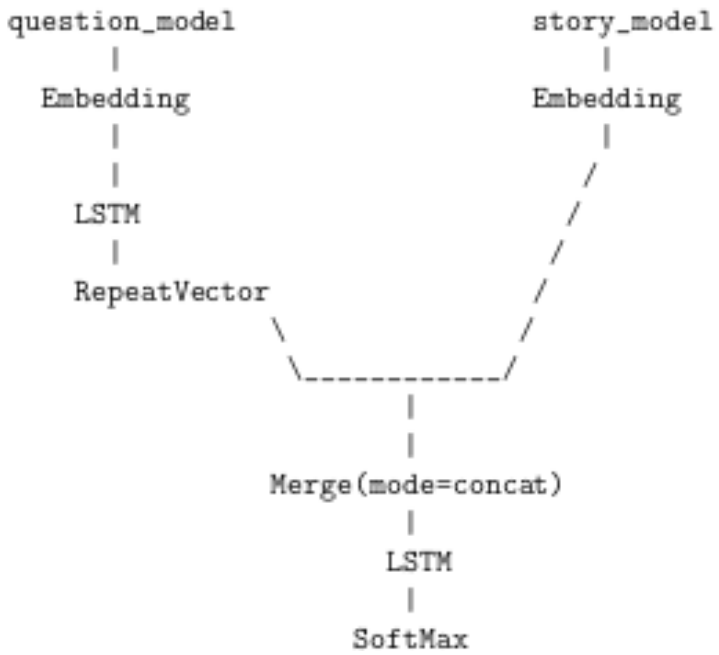
Modèle 1

Une première approche consistait en un modèle semblant plutôt naturel, diviser le modèle en deux (une partie histoire et une partie question) en appliquant des couches Embeddings et des couche LSTM. Il s'agira du modèle 1.



Modèle 2

Nous utiliserons également un autre modèle légèrement différent, repris de l'article[1]. Nous nommerons ce modèle par la suite : modèle 2.



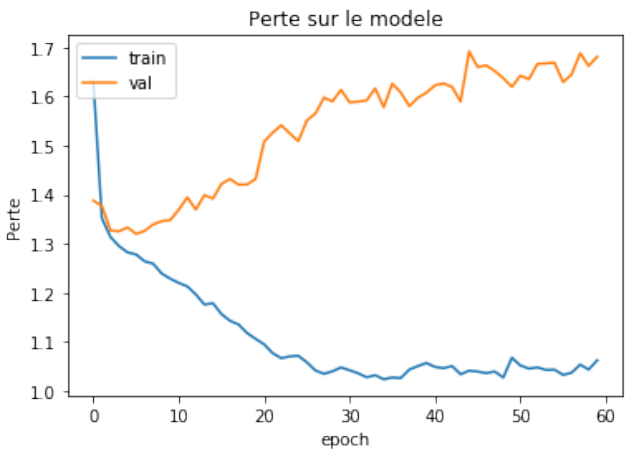
Modèle : Un problème de généralisation ?

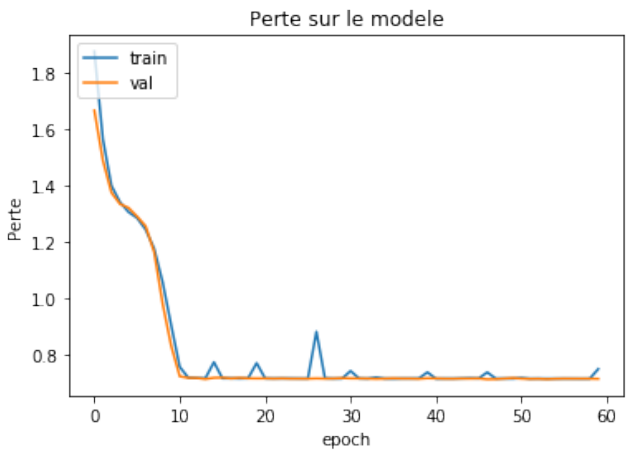
Rapidement nous voyons que le modèle 1 est bien moins performant que le modèle 2. Nous calculons les performances des deux modèles sur la première tâche avec la fonction **evaluate** de Keras sur deux ensembles de données de taille différente.

Modèle	Précision (1000 samples)	Précision (5500 samples)
Modèle 2	48%	66%
Modèle 1	36%	37%

Modèle : Un problème de généralisation ?

Nous observons par la suite grâce aux **courbes d'apprentissage** que le modèle 1 souffre d'un sur-apprentissage.





La différence est marquante, on voit pour le modèle 1 que la perte sur l'ensemble de validation augmente **fortement** à mesure que la perte sur l'ensemble d'apprentissage diminue : caractéristique d'un sur-apprentissage.

Résultats

Nous utilisons le modèle 2 pour nous résultats,

Quelques exemples de résultats,

Tache	Perte	Précision
1 Single Supporting Fact	1.19	51%
2 Two Supporting Facts	1.781	28%
3 Three Supporting Facts	1.718	19%
4 Two Argument Relations	1.458	35%
5 Three Argument Relations	1.183	39%
6 Yes/No Questions	0.697	48%
7 Counting	0.720	68%

Ces résultats sont obtenus en prenant en compte le bruit des données, avec le même modèle on obtient des résultat bien meilleurs sans le bruit (100% sur la première tâche)

Objectifs d'un second sujet

- Tester et modifier notre précédent système dans un autre contexte
- Changer le type d'inférence
- Identifier les limites

Présentation du sujet

Automatiser la notation de réponses courtes à des questions courtes

Présentation du sujet

Automatiser la notation de réponses courtes à des questions courtes

- Répondre au besoin de correcteurs dans le cadre d'un grand nombre de réponses
- Assister les étudiants en groupes réduits ou individuels

Descriptions des données

- 80 questions portant sur les sciences informatiques
- 31 élèves de niveaux différents
- 2273 réponses (car certaines questions peuvent être laissée sans réponse)

Description des données

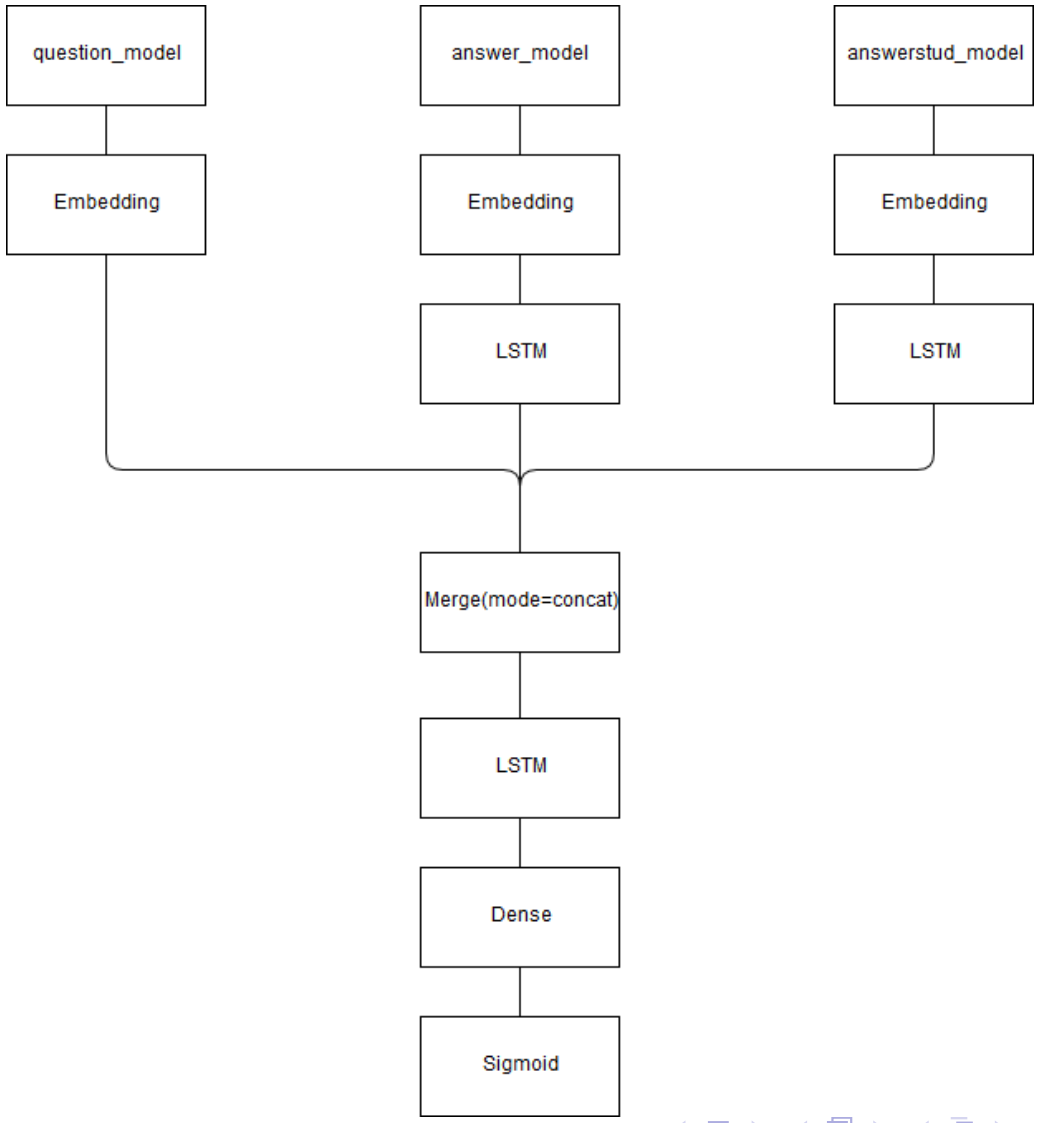
	Sample questions, correct answers, and student answers	Grades
Question:	What is the role of a prototype program in problem solving?	
Correct answer:	To simulate the behavior of portions of the desired software product.	
Student answer 1:	A prototype program is used in problem solving to collect data for the problem.	1, 2
Student answer 2:	It simulates the behavior of portions of the desired software product.	5, 5
Student answer 3:	To find problem and errors in a program before it is finalized.	2, 2
Question:	What are the main advantages associated with object-oriented programming?	
Correct answer:	Abstraction and reusability.	
Student answer 1:	They make it easier to reuse and adapt previously written code and they separate complex programs into smaller, easier to understand classes.	5, 4
Student answer 2:	Object oriented programming allows programmers to use an object with classes that can be changed and manipulated while not affecting the entire object at once.	1, 1
Student answer 3:	Reusable components, Extensibility, Maintainability, it reduces large problems into smaller more manageable problems.	4, 4

Nouvelles contraintes

Adapter notre modèle

- 3 données en entrée au lieu de 2
- Activation par fonction sigmoid
- Entraînement du modèle via fonction de coût *Minimum Squared Error*

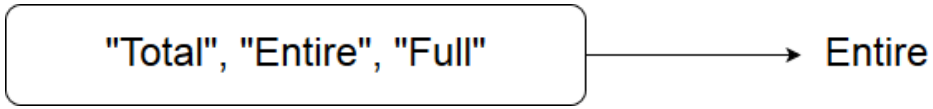
Modèle



Optimisation du vocabulaire

Idée pour améliorer l'apprentissage : Réduction du nombre de mots dans le vocabulaire On regroupe les mots par synonymes,

les mots appartenant à un même groupe de synonymes partagent un même indice



Optimisation du vocabulaire

Plusieurs solutions envisagées :

- Word2Vec
- Natural Language Toolkit (nltk)

Word2Vec n'étant pas le plus simple et adapté pour les synonymes, nous retiendrons nltk.

Optimisation du vocabulaire

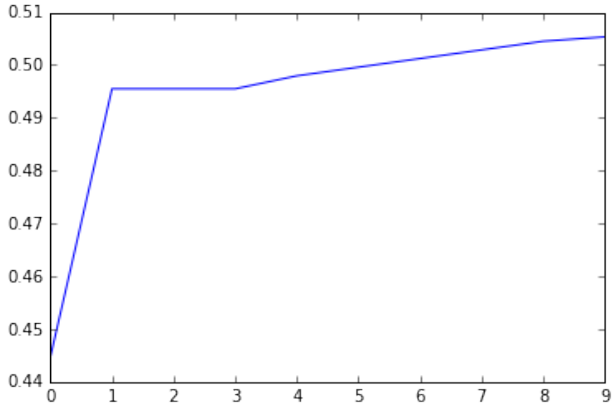
Comment fonctionne nltk :

On récupère tous les groupes de synonymes, les Synsets. Les mots d'un même synset partagent un sens commun.
Puis un récupère tous les lemmes de tous les synsets, avant de supprimer les doublons

Pour accéder aux synsets, nous avons utilisé le corpus Wordnet

Résultats

Précision avant regroupement de 50%



Seulement 52% environ après regroupement, en diminuant la taille du vocabulaire de 2200 mots à 1200.
Pour comparaison, les résultats dans l'article étaient plus proches des 80%

Conclusion

bAbI Tasks

- Implémentation en accord avec celle de l'article[1]
- Résultats très corrects et similaires à ceux obtenus dans l'article[1]
- Répond à l'objectif d'inférer un mot en modélisant un énoncé
- Différence étonnante entre les deux modèles présentés
- $\rightarrow$ Un système intéressant permettant d'imiter un raisonnement humain, vers un système général de raisonnement artificiel
- . . . mais réduit à utiliser un vocabulaire borné dans ce cas

Notation Automatique

Notre modèle n'est pas suffisamment adapté pour traiter ce sujet.

- Vocabulaire toujours trop grand ?
- Optimiser avec d'autres méthodes ?
- Taille d'embedding trop faible ?

Notation Automatique

Propositions :

- Travailler d'avantage sur la similarité entre les phrases avec Word2Vec (usage plus judicieux que les synonymes)
- Regrouper encore plus les mots avec du Stemming
- Utiliser nltk dans le cadre des antonymes, hyponymes, hyperonymes

Références I

Jason Weston, Antoine Bordes, Sumit Chopra, Alexander M. Rush, Bart van Merriënboer, Armand Joulin, Tomas Mikolov

Towards AI-Complete Question Answering : A Set of Prerequisite Toy Tasks

M. Mohler, R. Bunescu, R. Mihalcea

Learning to Grade Short Answer Questions using Semantic Similarity Measures and Dependency Graph Alignments