ELiRF-UPV at SemEval-2018 Task 11: Machine Comprehension using Commonsense Knowledge

José-Ángel González, Lluís-F. Hurtado, Encarna Segarra, Ferran Pla
Universitat Politècnica de València
Camí de Vera sn, 46022, València
{jogonba2|lhurtado|esegarra|fpla}@dsic.upv.es

Abstract

This paper describes the participation of ELiRF-UPV team at task 11, Machine Comprehension using Commonsense Knowledge, of SemEval-2018. Our approach is based on the use of word embeddings, NumberBatch Embeddings, and a Deep Learning architecture to find the best answer for the multiple-choice questions based on the narrative text. The results obtained are in line with those obtained by the other participants and they encourage us to continue working on this problem.

1 Introduction

In the Machine Comprehension using Commonsense Knowledge task, systems must answer multiple-choice questions given narrative texts about everyday activities. In addition to what is mentioned in the text, a substantial number of questions require inference using script knowledge about different scenarios.

In order to capture some script knowledge we decided to use a word representation based not only on distributional semantics word models but also on a knowledge graph, ConceptNet (Speer et al., 2016). ConceptNet is a knowledge graph that connects words and phrases of natural language with labeled edges. It is designed to represent the general knowledge involved in understanding language. ConceptNet could be used in combination with sources of distributional semantics, particularly the word2vec Google News skip-gram embeddings (Mikolov et al., 2013)) and GloVe 1.2 (Pennington et al., 2014), to produce new embeddings, NumberBatch embeddings, with state-of-the-art performance across many word-relatedness evaluations (Speer and Lowry-Duda, 2017).

More specifically, NumberBatch is a list of semantic word vectors which contains a complex meaning of those terms, beyond containing only contextual information like other kinds of embeddings based on distributional semantics e.g. Word2Vec or Glove. These embeddings are obtained through a combination of Word2Vec and Glove embeddings with knowledge extracted from ConceptNet by means of a technique known as retrofitting (Faruqui et al., 2014).

In this work, we used word representations based on NumberBatch embeddings because these representations encode semantically rich information related to the commonsense. Moreover, in order to tackle this machine comprehension task, we used a Deep Learning architecture with new attention mechanisms. The inclusion of these new attention mechanisms allow us to better capture the similarities among the elements of the input. The attention mechanisms we introduce in this work are suggested in the work (Seo et al., 2016), that obtained very competitive results in Question Answering tasks.

2 Resources and Preprocess

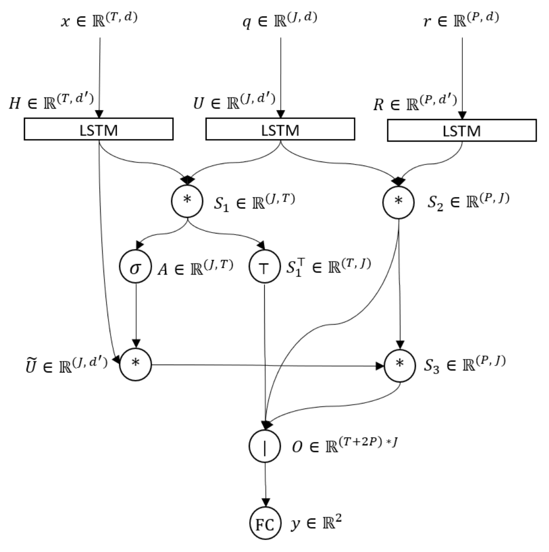
As we pointed in Section 1, NumberBatch embeddings were used for the representation of words. These embedding are provided by ConceptNet 5, which was compiled by the Commonsense Computing Initiative. ConceptNet 5 is freely available under the Creative Commons Attribution-ShareAlike license (CC BY SA 4.0) from http://conceptnet.io.

We explored several preprocessing techniques in the development phase The best results were obtained with the following preprocess: the conversion of all text to lowercase and the elimination of the question marks "?". After this, we carried out a tokenization process.

3 System Description

We tested Deep Learning architectures based on similarities between d-dimensional NumberBatch embeddings of story (x), question (q) and answer (r). Specifically, our approaches learn representations of x, q and r to first compute similarities, and then make a classification decision.

These kind of systems work well in Question Answering tasks, for instance, BiDAF (Seo et al., 2016) or QA-LSTM-Story(Pal and Sharma, 2016). For this reason, with the aim of improving the accuracy of these systems for this task, we incorporated some attention mechanisms of BiDAF in the QA-LSTM-Story system. A scheme of our system is shown in Figure 1.



$x \in \mathbb{R}^{(T,d)}$
$q \in \mathbb{R}^{(J,d)}$
$r \in \mathbb{R}^{(P,d)}$
$H \in \mathbb{R}^{(T,d')}$
$U \in \mathbb{R}^{(J,d')}$
$R \in \mathbb{R}^{(P,d')}$
LSTM
LSTM
LSTM
$*$
$S_1 \in \mathbb{R}^{(J,T)}$
$*$
$S_2 \in \mathbb{R}^{(P,J)}$
σ
$A \in \mathbb{R}^{(J,T)}$
$\top$
$S_1^\top \in \mathbb{R}^{(T,J)}$
$\widetilde{U} \in \mathbb{R}^{(J,d')}$
$*$
$*$
$S_3 \in \mathbb{R}^{(P,J)}$
$|$
$O \in \mathbb{R}^{(T+2P)*J}$
FC
$y \in \mathbb{R}^2$

Figure 1: System architecture.

All Deep Learning systems tested in this work take first a story $(x \in \mathbb{R}^{\{T,d\}})$, a question $(q \in \mathbb{R}^{\{J,d\}})$ and an answer $(r \in \mathbb{R}^{\{P,d\}})$ as input. Specifically, each of these elements is a matrix with their word embeddings as rows. Note that the length of the representations (T, J and P) is fixed by adding zero padding at the beginning to reach the length of the longest element.

Second, x, q and r are processed by means of three non-shared Bidirectional Long Short Term Memory (BLSTM) (Hochreiter and Schmidhuber, 1997) (Schuster and Paliwal, 1997). These networks capture useful features (X, Q and R) to make decisions based on similarities among the inputs. Moreover, we used BatchNormalization (Ioffe and Szegedy, 2015) and Dropout (Srivastava et al., 2014) with $p = 0.3$, after the input layer and after the BLSTM output to improve the generalization of the model.

After that, we compute similarities between each term of Q and each term of X ($S_1 = QX \in \mathbb{R}^{\{J,T\}}$) and similarities between each term of R and each term of Q (in a similar way, $S_2 = RQ \in \mathbb{R}^{\{P,J\}}$).

Now, if we concatenate S_1 and S_2, and we apply a fully-connected layer with softmax activation functions to classify, we reproduce exactly the QA-LSTM-Story system. However, in this work, we incorporated several attention mechanisms to this architecture in order to learn more complex relationships among the inputs.

One of these attention mechanisms is an adaptation of BiDAF. This adaptation used, in addition to Query2Context (Q2X) and Context2Query (X2Q), two additional attention mechanisms, Answer2Query (R2Q) and Query2Answer (Q2R). R2Q and Q2R are identical to X2Q and Q2X but applied to the question q and the answer r.

We have also tested many other attentions, but the best system obtained in the development phase only contains X2Q. In order to obtain this attention, we first transform the similarities S_1 by applying a softmax activation function to each row i.e. $A[i,j] = \frac{e^{S_1[i,j]}}{\sum_{t=0}^{T} e^{S_1[i,t]}}$. After this, we compute $\widetilde{Q} = AX$, to represent each row of Q as a weighted sum of the rows in X. That is, each row of Q is adapted in order to consider the most relevant rows of X.

From $\widetilde{Q}$, we can consider more explicit relationships between x and r if we compute the similarities between $\widetilde{Q}$ and R, in the same way as S_1 and S_2, to obtain S_3.

With all, we transpose the matrix S_1 to later concatenate all the similarity matrices. The result, ($S_1^\intercal$, S_2, S_3) $\in \mathbb{R}^{\{(T+2P),J\}}$, is flattened by concatenating all rows as columns to obtain a vector $O^{(T+2P)*J}$. Finally, we apply a softmax fully-connected layer to O to carry out the classification.

To train the system, we generated a training set consisting of all the triples of the corpus (x, q, r). Thus, for each x and q, we generate two triples, (x, q, r_1) and (x, q, r_2). Then, if r_1 is correct, $y(x,q,r_1) = 1$, else $y(x,q,r_1) = 0$. At inference time, given x, q and their two possible answers r_1 and r_2, we first build two triples (x, q, r_1) and (x, q, r_2) and, second, we obtain the network outputs y_1 and y_2, respectively. Finally, in order to decide which answer is correct, we select the one

Sys.	System Description	Acc. (%)
1	QA-LSTM-Story	79.21
2	BiDAF Adaptation	76.47
3	QA-LSTM-Story with X2Q	80.08

Table 1: Results on the development set.

Team	Acc. (%)
Yuanfudao (1/11)	83.95
Mitre (2/11)	82.27
Jiangnan (3/11)	80.91
ELiRF-UPV (4/11)	74.97
YNU_Deep (5/11)	74.72
...	
IUCM (11/11)	61.35

Table 2: Official results on the test set.

that maximizes the output for the correct class i.e. $y_i[1]$ (2nd component of the network output for a triple i).

4 Experimental Results

The results obtained during the development phase for the different systems above mentioned are shown in the Table 1. It can be observed that Deep Learning systems with simpler attention mechanisms worked better than those with more complex attention mechanisms (system 2 versus systems 1 and 3). If X2Q was added to compute more explicit relations between x and r, the accuracy slightly improved from 79.21% to 80.08% (system 1 versus system 3). Moreover, we tested system 3 with word2vec (Google News skip-gram) instead of NumberBatch embeddings obtaining an accuracy of 78.84%.

With these results, we chose the best system in the development phase (system 3 in Table 1).

The results obtained with this system on the test set are shown in Table 2.

5 Analysis of Results

Now, we make an analysis of the results obtained with our best system. In particular, we analyze the network confidence at intervals when deciding what is the correct answer (r_1 or r_2) given a story x and a question q. This confidence c is defined as the absolute difference between the outputs for the correct class of each answer ($y_1[1]$ and $y_2[1]$) i.e. $c = |y_1[1] - y_2[1]|$.

During this analysis, we get a maximum confidence $\underset{max}{c} = 0.999$ and a minimum confidence $\underset{min}{c} = 0.000$. Thus, there are extreme cases where the system is totally sure about what is the correct answer, or has total uncertainty. In the instance ($ix = 235$, $iq = 1$), we observe that the system is totally sure about the correct answer due to the answer has been explicitly found in the story. In a second instance ($ix = 131$, $iq = 4$), the system has total uncertainty because “\$20.00” and “\$15.00” do not appear in the NumberBatch embeddings. (Where ix and iq refers to the index of the story and the question in the test set).

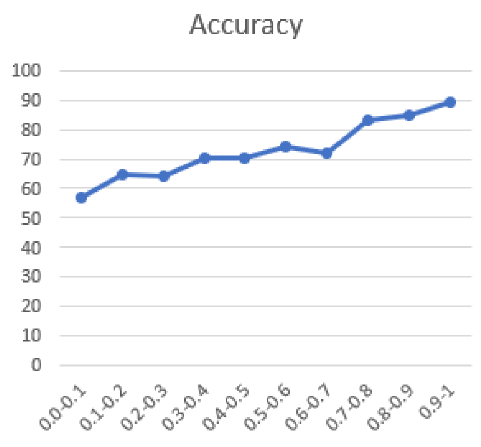


Figure 2: Accuracy in each confidence interval.

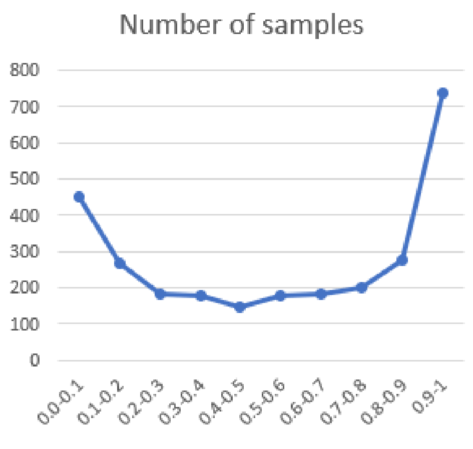


Figure 3: Number of samples in each confidence interval.

Moreover, we have also performed a study on the system accuracy and the number of samples for each confidence level between [0, 1]. The results obtained are shown in the Figures 2 and 3.

In general, as it can be observed in Figure 2, the greater the confidence, the better results were obtained. However, in Figure 3, we observed that there are many samples with very low confidence values, e.g. 0.0-0.1. We think that in order to reduce the number of samples in this confidence interval, it would be necessary to incorporate new knowledge resources.

6 Conclusions

In this work, we presented a Deep Learning architecture with new attention mechanisms in order to learn more complex representations and similarities among input elements (story x, question q and answer r). In order to capture some script knowledge, NumberBatch embeddings were used for the representation of words. With this approach we obtained competitive results.

As future work, we propose the study and development of new attention mechanisms to learn complex features and relationships. Moreover, we also find interesting the enrichment of the Deep Learning architectures with some commonsense information beyond the use of NumberBatch embeddings, such as the script knowledge resources suggested by the competition organizers.

7 Acknowledgements

This work has been partially supported by the Spanish MINECO and FEDER founds under projects ASLP-MULAN: Audio, Speech and Language Processing for Multimedia Analytics (TIN2014-54288-C4-3-R); and AMIC: Affective Multimedia Analytics with Inclusive and Natural Communication (TIN2017-85854-C4-2-R). Work of José-Ángel González is also financed by Universitat Politècnica de València under grant PAID-01-17.

References

Manaal Faruqui, Jesse Dodge, Sujay Kumar Jauhar, Chris Dyer, Eduard H. Hovy, and Noah A. Smith. 2014. Retrofitting word vectors to semantic lexicons. *CoRR*, abs/1411.4166.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long Short-Term Memory. *Neural Computation*, 9(8):1735–1780.

Sergey Ioffe and Christian Szegedy. 2015. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *CoRR*, abs/1502.03167.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Distributed representations of words and phrases and their compositionality. *CoRR*, abs/1310.4546.

Sujit Pal and Abhishek Sharma. 2016. Deep learning models for question answering. Elsevier Search Guild Question Answering Workshop.

Jeffrey Pennington, Richard Socher, and Christopher D Manning. 2014. Glove: Global vectors for word representation. In *EMNLP*, volume 14, pages 1532–1543.

M. Schuster and K.K. Paliwal. 1997. Bidirectional recurrent neural networks. *Trans. Sig. Proc.*, 45(11):2673–2681.

Min Joon Seo, Aniruddha Kembhavi, Ali Farhadi, and Hannaneh Hajishirzi. 2016. Bidirectional attention flow for machine comprehension. *CoRR*, abs/1611.01603.

Robert Speer, Joshua Chin, and Catherine Havasi. 2016. Conceptnet 5.5: An open multilingual graph of general knowledge. *CoRR*, abs/1612.03975.

Robert Speer and Joanna Lowry-Duda. 2017. Conceptnet at semeval-2017 task 2: Extending word embeddings with multilingual relational knowledge. *CoRR*, abs/1704.03560.

Nitish Srivastava, Geoffrey E. Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: a simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(1):1929–1958.