FA3L at SemEval-2017 Task 3: A ThRee Embeddings Recurrent Neural Network for Question Answering

Giuseppe Attardi, Antonio Carta, Federico Errica, Andrea Madotto, Ludovica Pannitto
Department of Computer Science, University of Pisa
Largo B. Pontecorvo, 3
attardi@di.unipi.it, a.carta@outlook.it, f.errica@protonmail.com
andreamad8@gmail.com, ellepannitto@gmail.com

Abstract

In this paper we present ThReeNN, a model for Community Question Answering, Task 3, of SemEval-2017. The proposed model exploits both syntactic and semantic information to build a single and meaningful embedding space. Using a dependency parser in combination with word embeddings, the model creates sequences of inputs for a Recurrent Neural Network, which are then used for the ranking purposes of the Task. The score obtained on the official test data shows promising results.

1 Introduction

Community Question Answering (cQA) systems have proven to be useful for a long time and they still are an invaluable source of information. However, due to their rapid growth and to the large amount of data provided it is not easy to find a relevant answer or a good related question amongst all the others. For these reasons we present a model which tries to tackle these problems. The subtasks we have worked on can be described as follows:

A) Question-Comment Similarity - Given a question q and 10 comments $c_1, \ldots, c_{10}$, rank such comments from the most relevant to the least one with respect to q, and assign to each one a label which can be "Good" or "Bad".

B) Question-Question Similarity - Given a question q and a set of 10 related questions $q_1, \ldots, q_{10}$, rank the 10 questions from "Relevant" to "Irrelevant", according to q.

A more detailed description of the task can be found in (Nakov et al., 2017).

Our work has been inspired by studies regarding embedding spaces. Indeed, in (Hsu et al., 2016) GloVe embeddings (Pennington et al., 2014) are used to solve the same subtasks as ours, achieving good results using just word embeddings which encode semantic information into a vector. Moreover, the model proposed in (Yu et al., 2013), where autoencoders are used to build an embedding space, has been exploited to propose an approach that mixes semantic and syntactic information through the use of word embeddings and dependency parsing. These are then put together and become an input for the neural network. In this way we try to enhance the capability of the learning system.

In principle, our approach aims at enriching semantic information with syntactic relations holding between elements of the couples (question-comment or question-question). This should serve well for both subtasks A and B, since the model will learn relations between a question and a comment or between a question and another one. However, further research would be useful to understand to what extent there exist differences in the kind of relations learnt, and therefore in the subtasks.

The paper is organised as follows: Section 2 outlines the preprocessing and additional features used by the model, while Section 3 describes the key models used. Section 4 shows the model selection strategy and the alternatives we explored with respect to word embeddings and their combination. Finally, Section 5 reports performances on different models and Section 6 wraps up everything and discusses about future works. From now on, we will refer to "comment" for indicating both a comment (Subtask A) or a related question (Subtask B), since our model does not make distinctions between them. We participated to Semeval 2017, ranking 8th in Subtask A and 10th in Subtask B.

Proceedings of the 11th International Workshop on Semantic Evaluations (SemEval-2017), pages 299–304,
Vancouver, Canada, August 3 - 4, 2017.

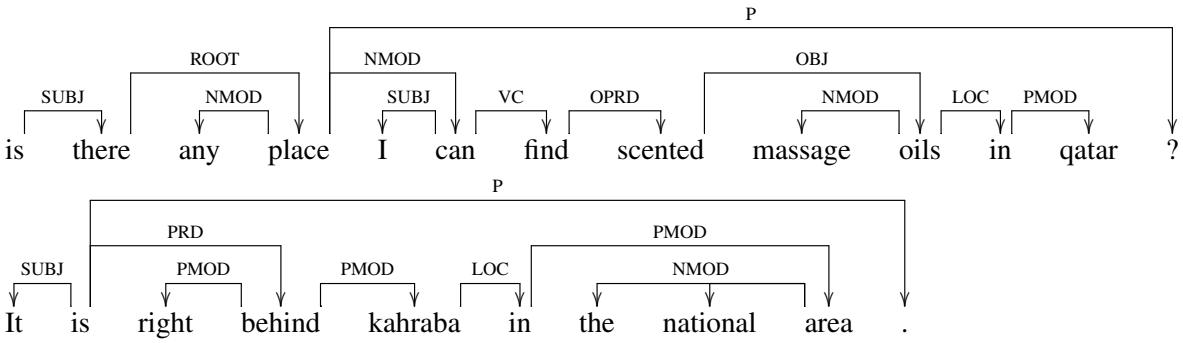

Figure 1: Dependency parsing of two sentences taken from a question and a comment in the training set. In this example the first input $x(t)$ of the RNN is going to be: <"is",SUBJ,"there","is",SUBJ,"It">.

2 Data Preprocessing

We applied standard preprocessing to question and comment body, so as to achieve better performance during syntactic parsing and a better alignment of our vocabulary to the GloVe one. Each question also includes the subject of the topic. Preprocessing included the following steps:

- Portions of text that include HTML tags and special sequences were removed or substituted with simpler strings.
- Using a set of regular expressions, we replaced URLs, nicknames, email addresses with a placeholder for each category.
- Too long repetitions of characters inside tokens were replaced by a single character (e.g. *loooot* became *lot*). Indeed, in the language spoken on community forums, letters are often repeated to emphasize words; with our approach we were able to reconstruct their standard form. Moreover, multiple punctuation was also collapsed.
- Standard use of spacing after punctuation was restored, in order to avoid problems during tokenization.
- Using a hand-written dictionary, the most common abbreviations were replaced with the corresponding extended form.

We then performed sentence splitting and tokenization using *nltk* (Bird et al., 2009). During the tokenization step, we performed spelling corrections.

Finally, texts were analyzed using Tanl pipeline (Attardi et al., 2007), adding morpho-syntactic and syntactic information (i.e., part of speech tagging and dependency parsing). Figure 1 shows an example of a question and a comment which are parsed accordingly.

2.1 Additional Features

After that, we generated several features, representing both metadata and some properties of the couple Question-Comment. These features have been commonly used in literature, both with Neural Networks (as in (Mohtarami et al., 2016)), linear or SVM models as in (Mihaylova et al., 2016), in order to include additional and potentially relevant information not easily conveyed through semantic representations. In our case, they are used as additional input beside the RNN output. Features can be grouped as follows:

- Features encoding information about standard similarity between question and comment (all measures are expressed in terms of number of tokens):
  - size of intersection between question and comment
  - Jaccard Coefficent (ratio between intersection size and union size of question and comment)
  - comment length
  - ratio between comment length and question length
  - length of the longest common subsequence between question and comment
- Features encoding metadata information, in particular:
  - number of the comment in default ordering

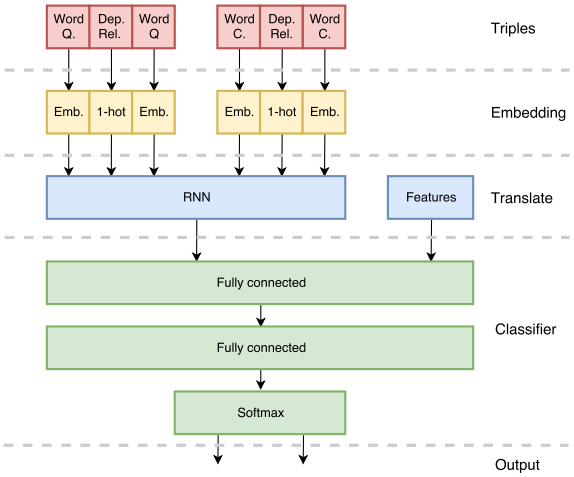

Figure 2: Conceptual schema of the model used for the classification.

  - whether the comment was posted by the same user asking the question
  - whether the user posting the comment had already posted a comment for the same question

- Features encoding presence of certain elements in comment body, in particular we looked for:
  - presence of question marks
  - presence of URLs (through regex)
  - presence of username (through regex)
  - presence of a username among those that are authors of comments preceeding the considered one

3 Model

The proposed model[1] makes use of the previous steps (i.e. a dependency parser) whose output is a tree, to generate a sequence of triples. The ith triple is made of $< W_i, rel, W_r >$, where W_i is the ith word of the text and W_r is the word associated through rel (i.e. the dependency relation extracted by the parser). Then triples $< e_i, rel, e_r >$ are generated, where e_i and e_r are word-embeddings vectors for the two words, and rel is a 1-hot-encoding of the dependency relations. The kth input to be fed to the RNN is simply made by concatenating the kth embedding triple of the comment with the kth one of the question. Figure 1 shows an example of how to obtain a valid input for our model. Our goal is to let the system learn the correct composition rule through syntactic dependencies.

Hence, the input of our model is dual: a sequence of triples which represents the question and another sequence for the comments. These are then passed to a sentence encoder, which is a Recurrent Neural Network (RNN), that is used to return a single output aiming to represent the entire sequences. In particular we describe a Long Short Term Memory (Hochreiter and Schmidhuber, 1997) which are capable of learning long-term dependencies; Then, given x as input in the form:

$$x^{(t)} = < e_{Qi}^{(t)}, Rel_Q^{(t)}, e_{Qr}^{(t)}, e_{Ci}^{(t)}, Rel_C^{(t)}, e_{Cr}^{(t)} >$$

we have:

$$\begin{aligned}
f^{(t)} &= \sigma\left(U^f x^{(t)} + W^f h^{(t-1)} + b^f\right) \\
g^{(t)} &= \sigma\left(U^g x^{(t)} + W^g h^{(t-1)} + b^g\right) \\
y^{(t)} &= \sigma\left(U^i x^{(t)} + W^i h^{(t-1)} + b^i\right) \\
s^{(t)} &= f^{(t)} \odot s^{(t-1)} + g^{(t)} \odot y^{(t)} \\
o^{(t)} &= \sigma\left(U^o x^{(t)} + W^o h^{(t-1)}\right) \\
h^{(t)} &= tanh(s^{(t)}) \odot o^{(t)}
\end{aligned}$$

where $f^{(t)}$ is the forget gate, $g^{(t)}$ the input gate, $s^{(t)}$ the state, $o^{(t)}$ the output gate and $h^{(t)}$ the hidden state. U and W are the weight matrices for each gate (e.g., U^o refers to the matrix for the output gate) and $\odot$ is the Hadamard product. Then the RNN output, along with a vector made up of additional features, become the inputs passed to the final feed-forward layers which performs the scoring.

Each layer of the final network uses a sigmoid activation function. Hence, given x, the layer input, W and b the layer matrix and bias, the output y is defined as

$$y = \sigma(Wx + b)$$

The final output o of the network uses a softmax activation, thus we have:

$$o_i = \frac{exp(y_i)}{\sum_{j=0}^{n} exp(y_j)}$$

Where n is the length of the vector y. The latter provides a distribution over two classes: 'Good'

[1] An implementation is available at https://github.com/AntonioCarta/ThreeRNN

and 'Bad/Partially Useful' for subtask A, 'PerfectMatch/Relevant' and 'Irrelevant' for subtask B. To obtain the final ranking we took the probability of a given input to be labeled as the positive class. The entire network is trained with back-propagation using a cross-entropy loss function. Figure 2 shows the conceptual schema of the model.

4 Experiments

To perform model selection we merged *training* and *development* files provided by Semeval organisers, then we shuffled and extracted a training and a validation set. We selected various hyper-parameters, shown with their values in Table 4, such as learning rate, number of hidden units and hidden layers for the recurrent and feedforward layers, dropout (Srivastava et al., 2014), L2 regularization, activation functions (i.e. ReLu (Nair and Hinton, 2010), sigmoid and hyperbolic tangent), optimization algorithms (i.e. adam (Kingma and Ba, 2014) and rmsprop (Tieleman and Hinton, 2012)). The length threshold for the number of triples in input to the RNN as been also added as hyper-parameter (i.e., Max length); if the comment/question is shorter, it is filled up with zeros ("null triples"). Since each training required quite a large amount of time, we opted for a random search technique (Bergstra and Bengio, 2012).

Parameter	Values
RNN	LSTM, **GRU**, SUM
RNN layer	1,**2**
Hidden layer	1,2,**3**
Embeddings size	**100**, 200, 300
Hidden size	**50**, 100, 200
Max length	5,**10**,25,50,75,100,150
Dropout	0, **0.1**, 0.2, 0.3, 0.4
L2	0.01, **0.001**, 0.0001, 0.00001
Activation	ReLu, tanh, **sigmoid**
Optimizer	**adam**, rmsprop

Table 1: Hyper-parameters used during model selection. The selected parameters for Subtask A are in bold, and underlined for Subtask B.

The embeddings layer uses pretrained embeddings which are fixed during the training phase. We tried to update them together with the entire network during training but the resulting network always ended up to over-fit. Two different types of embeddings have been evaluated: GloVe (Pennington et al., 2014), which are trained using Wikipedia, and embeddings trained directly with questions and answers extracted from the Qatar Living forum (Mihaylov and Nakov, 2016). However, in our model both embeddings worked well, thus with the latter we did not obtained any particular improvements.

To encode the RNN input into a single embedding we compare three different approaches: SUM (which sums all the triples given as input), LSTM (Hochreiter and Schmidhuber, 1997) and GRU (Cho et al., 2014).

Finally, the neural network model was implemented using Keras (Chollet, 2015), which provides an efficient and easy-to-use deep learning utilities.

5 Results

The results obtained in the test set, in both sub-task A and B, are summarized in Table 2. The primary submission uses LSTM for subtask A and GRU for subtask B. Instead, the contrastive model uses SUM as aggregation and it has been submitted just for the subtask A. Using the SUM model, which is computationally less expensive than RNN, we obtained just a slightly worst MAP (i.e. around 0.5%), which suggests we could further improve the performance by making the RNN exploit better the sequence in input. Moreover, there is a trade-off between representation length and computational costs, achieved with the use of the length threshold; this may be regarded as a crucial choice for our model.

	Subtask A		**Subtask B**	
	MAP	Acc	MAP	Acc
Baseline (IR)	72.61	-	41.85	-
Primary	83.42	68.02	42.24	73.86
Contrastive	82.87	68.67	-	-

Table 2: Summary of the results of the submitted model on subtask A and B

6 Conclusions

To sum up, we have developed a model which tries to combine semantic and syntactic information into a single vector space. We will further investigate this combination, through the use of

syntactic relations holding between content words, rather than exploiting the whole set of dependency relations (e.g. different tag-sets, partial or shallow parsing of sentences etc.). Our experiments have explored different possibilities regarding the choice of the word embedding system; all of them proved in the end to achieve similar results. However, it may be worth trying to build an ad-hoc embedding space which mixes parsing and lexical information, aiming to improve the performances of our model. Future works may include improvements to the RNN in order to better represent longer sentences, or the use of recursive neural network that directly use the tree structure given by the dependency parsing, with different weights matrices for each dependency relation.

References

Giuseppe Attardi, Felice Dell'Orletta, Maria Simi, Atanas Chanev, and Massimiliano Ciaramita. 2007. Multilingual dependency parsing and domain adaptation using desr. In *EMNLP-CoNLL*. pages 1112–1118.

James Bergstra and Yoshua Bengio. 2012. Random search for hyper-parameter optimization. *Journal of Machine Learning Research* 13(Feb):281–305.

Steven Bird, Ewan Klein, and Edward Loper. 2009. *Natural language processing with Python: analyzing text with the natural language toolkit*. " O'Reilly Media, Inc.".

Kyunghyun Cho, Bart Van Merriënboer, Dzmitry Bahdanau, and Yoshua Bengio. 2014. On the properties of neural machine translation: Encoder-decoder approaches. *arXiv preprint arXiv:1409.1259* .

François Chollet. 2015. Keras. `https://github.com/fchollet/keras`.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation* 9(8):1735–1780.

Wei-Ning Hsu, Yu Zhang, and James Glass. 2016. Recurrent neural network encoder with attention for community question answering. *arXiv preprint arXiv:1603.07044* .

Diederik Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980* .

Todor Mihaylov and Preslav Nakov. 2016. Semanticz at semeval-2016 task 3: Ranking relevant answers in community question answering using semantic similarity based on fine-tuned word embeddings. In *Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval 2016)*. Association for Computational Linguistics, San Diego, California, pages 804 – 811. http://www.aclweb.org/anthology/S16.

Tsvetomila Mihaylova, Pepa Gencheva, Martin Boyanov, Ivana Yovcheva, Todor Mihaylov, Momchil Hardalov, Yasen Kiprov, Daniel Balchev, Ivan Koychev, Preslav Nakov, Ivelina Nikolova, and Galia Angelova. 2016. Super team at semeval-2016 task 3: Building a feature-rich system for community question answering. In *Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval-2016)*. Association for Computational Linguistics, San Diego, California, pages 836–843. http://www.aclweb.org/anthology/S16-1129.

Mitra Mohtarami, Yonatan Belinkov, Wei-Ning Hsu, Yu Zhang, Tao Lei, Kfir Bar, Scott Cyphers, and Jim Glass. 2016. Sls at semeval-2016 task 3: Neural-based approaches for ranking in community question answering. In *Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval-2016)*. Association for Computational Linguistics, San Diego, California, pages 828–835. http://www.aclweb.org/anthology/S16-1128.

Vinod Nair and Geoffrey E Hinton. 2010. Rectified linear units improve restricted boltzmann machines. In *Proceedings of the 27th international conference on machine learning (ICML-10)*. pages 807–814.

Preslav Nakov, Doris Hoogeveen, Lluís Màrquez, Alessandro Moschitti, Hamdy Mubarak, Timothy Baldwin, and Karin Verspoor. 2017. Semeval-2017 task 3: Community question answering. In *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)*. Association for Computational Linguistics, Vancouver, Canada, pages 27–48. http://www.aclweb.org/anthology/S17-2003.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. Glove: Global vectors for word representation. In *Empirical Methods in Natural Language Processing (EMNLP)*. pages 1532–1543. http://www.aclweb.org/anthology/D14-1162.

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.

T Tieleman and G Hinton. 2012. Rmsprop: Divide the gradient by a running average of its recent magnitude. coursera: Neural networks for machine learning. Technical report, Technical report.

Wenchao Yu, Guangxiang Zeng, Ping Luo, Fuzhen Zhuang, Qing He, and Zhongzhi Shi. 2013. Embedding with autoencoder regularization. In *Joint Euro-*

pean Conference on Machine Learning and Knowledge Discovery in Databases. Springer, pages 208–223.