LCT-MALTA's Submission to RepEval 2017 Shared Task

Hoa Trong Vu, Thuong-Hai Pham, Xiaoyu Bai
Marc Tanti, Lonneke van der Plas, Albert Gatt
thuong-hai.pham.16@um.edu.mt, hoa.vutrong.16@um.edu.mt, xiaoyu.bai.16@um.edu.mt
marctanti@gmail.com, lonneke.vanderplas@um.edu.mt, albert.gatt@um.edu.mt
Institute of Linguistics, University of Malta, Malta

Abstract

We present in this paper our team LCT-MALTA's submission to the RepEval 2017 Shared Task on natural language inference. Our system is a simple system based on a standard BiLSTM architecture, using as input GloVe word embeddings augmented with further linguistic information. We use max pooling on the BiLSTM outputs to obtain embeddings for sentences. On both the matched and the mismatched test sets, our system clearly beats the shared task's BiLSTM baseline model.

1 Introduction

The RepEval 2017 Shared Task aims to evaluate fixed-length vector representations (or embeddings) of sentences on the basis of a natural language understanding task, viz. natural language inference (NLI), also known as recognising textual entailments. Given two sentences, the first being the premise and the second the hypothesis, the goal of NLI is to train a classifier to predict whether the relation of the hypothesis to the premise is one of entailment, contradiction or a neutral relation. The training and test data for this 3-way classification task at RepEval 2017 are drawn from the Multi-Genre NLI, or MultiNLI corpus (see Williams et al. (2017) for details). Task participants are provided with both training and development datasets, where parts of the development data match the training data in terms of genre, topic etc. (referred to as *matched* examples) and other parts do not (referred to as *mismatched* examples).

This paper presents Team LCT-MALTA's submission to the shared task. In line with previous research, we obtain a single vector which is the combined representation of both the premise and the hypothesis and feed it into a Multilayer Perceptron (MLP) for the actual 3-way classification.

2 Related Work

Various works in recent years have dealt with the creation of distributed sentence representations, typically based on existing word embeddings such as word2vec (Mikolov et al., 2013) or GloVe (Pennington et al., 2014). The baseline models at the shared task use GloVe vectors and present three approaches to obtaining sentence embeddings (Williams et al., 2017): a) taking the sum of the embeddings of all the words in the sentence (continous bag of words, CBOW); b) taking the average of the hidden state outputs of a bidirectional LSTM (BiLSTM; Hochreiter and Schmidhuber 1997) across all the words; and c) the Enhanced Sequential Inference Model (ESIM) by (Chen et al., 2017), which, however, relies on cross-sentence attention, which submissions to the shared task may not make use of.

Instead of the BiLSTM architecture, Tai et al. (2015) propose a tree-structured LSTM to capture the hierarchical structure of natural language sentences. Conneau et al. (2017) use BiLSTM with max pooling and achieve state-of-art results when testing their sentence representations on an NLI task based on the Stanford Natural Language Inference (SNLI) dataset (Bowman et al., 2015). Lin et al. (2017) introduce a self-attention mechanism with multiple hops of attention on top of BiLSTM, where the different hops attend to different parts of the input sentence. Their approach represents sentence embeddings as 2-D matrices instead of vectors.

3 Our Approach

We present in our submission a simple BiLSTM-based approach related to the second baseline model. Crucially, however, we add the following alterations:

- Our input word vectors are enhanced with part-of-speech (POS), dependency and word character information.
- Instead of taking the average of the BiLSTM outputs across all words (mean pooling), we use max pooling.

3.1 Enhanced Word Embeddings

A central component of our approach is the enhancement of the pre-trained GloVe vectors with additional linguistic information. The main motivation is the following: BiLSTM proceeds linearly, processing an input sentence word by word. The structure of natural language sentences, however, is hierarchical in nature [1]. We wish to encode some information on linguistic structure while keeping the simple, standard BiLSTM architecture. Therefore, we attach such additional information to the representations of individual words.

In conrete terms, we initialise our model with 300-D pre-trained GloVe vectors (as is done in the BiLSTM baseline model) and enhance them with the following content:

3.1.1 POS-tag Embeddings

Part-of-speech (POS) tagging is a common first step in syntactic sentence processing. We postulate that explicit knowledge of a the input words' syntactic categories might help representing the meaning of the input sentence. Thus, using modules from UDPipe (Straka et al., 2016), we tokenise and tag input sentences with universal POS-tags. We then generate randomly-initialised, 20-D embeddings for all POS-tags.

3.1.2 Dependency Label Embeddings

Dependency parsing captures the binary dependency relation between words in a sentence and determines the central word of an input sentence (the head word of the sentence) (Kübler et al., 2009). In particular, it is enable to encode longer dependencies across multiple words. As such, dependency parsing provides vital information on the sentence's structure.

Hence, we apply UDPipe's (Straka et al., 2016) state-of-the-art dependency parser to the input sentence and subsequently equip each pre-trained word embedding with that word's dependency information, which in turn consists of the word's head word and its dependency relation to the head. In concrete terms, for each word w_i, we map the embedding of its head word w_j to a 50-D vector and generate a 50-D randomly initialised embedding for the dependency relation from w_i to its head. We then take the element-wise product between these two 50-D vectors to obtain the full dependency embedding for the word w_i. Formally, the dependency embedding w_{i_d} for any token w_i is computed as follows:

$$w_{i_d} = E_d[d_{ij}] \odot (W_d * w_j) \quad (1)$$

where d_{ij} is the dependency relation between token w_i and the head token w_j, E_d is the 50-D embedding of that dependency relation, W_d is a matrix of size 50x300 which maps w_j (originally a 300-D GloVe vector) to a 50-D vector, and $\odot$ is the element-wise product.

3.1.3 Character Embeddings

The usage of character embeddings is mainly inspired by various works which incorporate character-level embeddings into the distributed representation of words to yield improved word embeddings (Santos and Zadrozny, 2014; Bojanowski et al., 2016; Kim et al., 2016). For each token, we employ LSTMs to compute embeddings for each of its characters. The LSTM input at each time step is one character, i.e. one letter, and the output is a 100-D hidden state. The last 100-D hidden state vector is then considered the full character embedding for that token.

3.1.4 Final Input Word Embeddings

Finally, for each word, we concatenate its original GloVe embeddings with all of the above-mentioned additional linguistic information. Thus, our final embedding for each word, which we input to the BiLSTM to compute sentence embeddings, consists of the concatenation of the word's 300-D Glove embedding, its 20-D POS-tag embedding, its 50-D dependency embedding and 100-D character embedding. As mentioned, all of the embeddings except for the GloVe

[1] As mentioned in section 2, Tai et al. (2015) propose a *tree*-structured LSTM for similar reasons

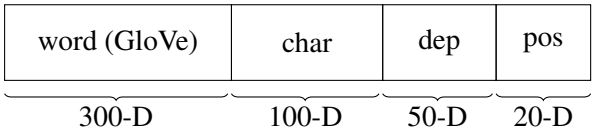

Figure 1: Our enhanced word embedding

word embeddings are randomly initialized. Figure 1 illustrates an instance of our final enhanced word embeddings.

3.2 BiLSTM Sentence Embedding with Max Pooling

The enhanced word embeddings described above are fed into a standard BiLSTM architecture, with a 100-D hidden state output vector for each unidirectional LSTM. Concatenating the forward ($\overrightarrow{h_t}$) and backward ($\overleftarrow{h_t}$) output vectors for each node, we obtain *n* hidden state vectors $\overrightarrow{h}$, where *n* is the number of words in the input sentence and $\overrightarrow{h}$ is a 200-D vector corresponding to one word.

$$\overrightarrow{h_t} = \overrightarrow{LSTM}(w_t, \overrightarrow{h_{t-1}}) \quad (2)$$

$$\overleftarrow{h_t} = \overleftarrow{LSTM}(w_t, \overleftarrow{h_{t-1}}) \quad (3)$$

Conneau et al. (2017) examine various supervised methods of obtaining general-purpose sentence embeddings, with their conclusion favouring a BiLSTM architecture combined with max pooling. We follow their findings and use max pooling on the *n* BiLSTM hidden outputs $\overrightarrow{h}$. Indeed, our own experiments shall show the superiority of max pooling over average pooling (see section 4). Hence, for each dimension *d* of the 200 dimensions, we take the maximum among the values of all hidden outputs at *d*. The result is a single 200-D vector embedding of the input sentence.

3.3 Combining Premise and Hypothesis Representations

We separately obtain the previously described sentence embeddings for the premise and the hypothesis. Mou et al. (2015) examine multiple heuristics for combining the same-length embeddings of two sentences in NLI tasks, including concatenating the two vectors and taking their element-wise difference or product. We use a combination of some of these heuristics. More specifically, we concatenate the two sentence embeddings, then further concatenate a) their element-wise maximum and b) their element-wise product

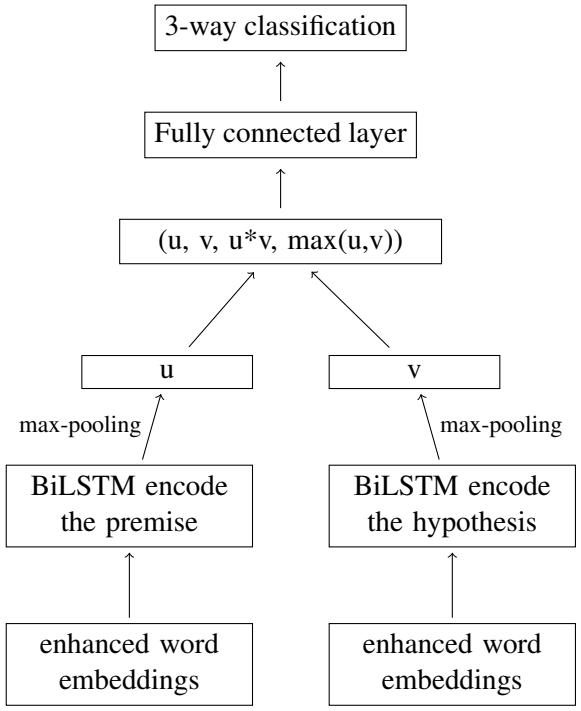

Figure 2: Architecture of our final submission

to the concatenation of the two sentence embeddings. Hence, we obtain as a result a single 800-D vector that is the combined representation of the premise and the hypothesis.

3.4 Classification

Finally, we feed the single vector representation of the two sentences to a *tanh* layer with a *softmax* layer on top of it to perform the 3-way classification to the classes *entailment*, *contradiction*, and *neutral*. Our complete model is illustrated in Figure 2.

4 Experiments & Results

We experimented with BiLSTM-based sentence encoders including and excluding our enhanced word embeddings as well as in combination with max pooling and average pooling. We use L2 regularization and set dropout rate to 0.1 to prevent overfitting. The models are trained in 10 epoches using Adam optimizer with learning rate 10^{-3}. The models having the best performance on development set are selected to evaluate on test set.

Furthermore, we implemented two systems presented in literature, viz. (Lin et al., 2017)'s self-attentive embeddings approach and (Kim, 2014)'s convolutional neural network (CNN) approach, and compared their performance with that of our

Method	Matched		Mismatched	
	Dev	Test	Dev	Test
BiLSTM + average pooling (baseline)	66.9	_	66.9	_
BiLSTM + max pooling	67.23	_	67.18	_
CNN (Kim, 2014)	67.3	_	68.0	_
Automatically learned self-attentive embeddings (Lin et al., 2017)	67.89	_	67.69	_
BiLSTM + enhanced embedding + average pooling	68.3	_	67.8	_
BiLSTM + enhanced embedding + max pooling	70.8	**70.7**	71.1	**70.8**

Table 1: System performances on MultiNLI dataset in order of ascending accuracy scores.

system. In their original work, Lin et al. (2017) use 30 hops of attention for each sentence. In our implementation of their algorithm, we deemed 10 hops to be sufficient due to the limited length of the sentences in our MultiNLI database.

The results of our experiments are shown in Table 1, in order of ascending accuracy scores. In our experiments, our final submission, i.e. the BiLSTM-encoder with enhanced word embeddings and max pooling, as described in section 3, performed best, obtaining accuracy scores of **70.8** and **71.1** on the matched and mismatched dev set, respectively. Its final results on the full shared task test set are **70.7** for matched and **70.8** for mismatched data.

5 Discussion

Our results favour max pooling over average pooling, which is in agreement with findings by Conneau et al. (2017). Moreover, our enhanced word embeddings are shown to be effective. Their addition alone produces the accuracy scores superior to what the incorporation of (Lin et al., 2017)'s automatically learned self-attention matrix yields. The combination of max pooling and enhanced word embeddings, which are extremely simple alterations to the BiLSTM baseline, yield results which clearly beat the baseline.

Thus, our submitted system to the RepEval 2017 shared task demonstrates that simple alterations to the standard BiLSTM architecture for computing sentence embeddings can obtain visible improvements. In particular, *linguistic* information is shown to be useful for the present NLI task. Therefore, with respect to distributed representations of sentence meaning, more sophisticated systems which take into account linguistic and grammatical relationships are worth further investigation.

References

Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2016. Enriching word vectors with subword information. *arXiv preprint arXiv:1607.04606* .

Samuel R Bowman, Gabor Angeli, Christopher Potts, and Christopher D Manning. 2015. A large annotated corpus for learning natural language inference. *arXiv preprint arXiv:1508.05326* .

Qian Chen, Xiaodan Zhu, Zhen-Hua Ling, Si Wei, Hui Jiang, and Diana Inkpen. 2017. Enhanced lstm for natural language inference. In *Proc. ACL*.

Alexis Conneau, Douwe Kiela, Holger Schwenk, Loic Barrault, and Antoine Bordes. 2017. Supervised learning of universal sentence representations from natural language inference data. *arXiv preprint arXiv:1705.02364* .

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

Yoon Kim. 2014. Convolutional neural networks for sentence classification. *arXiv preprint arXiv:1408.5882* .

Yoon Kim, Yacine Jernite, David Sontag, and Alexander M Rush. 2016. Character-aware neural language models. In *Thirtieth AAAI Conference on Artificial Intelligence*.

Sandra Kübler, Ryan McDonald, and Joakim Nivre. 2009. Dependency parsing. *Synthesis Lectures on Human Language Technologies* 1(1):1–127.

Zhouhan Lin, Minwei Feng, Cicero Nogueira dos Santos, Mo Yu, Bing Xiang, Bowen Zhou, and Yoshua Bengio. 2017. A structured self-attentive sentence embedding. *arXiv preprint arXiv:1703.03130* .

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*. pages 3111–3119.

Lili Mou, Rui Men, Ge Li, Yan Xu, Lu Zhang, Rui Yan, and Zhi Jin. 2015. Natural language inference

by tree-based convolution and heuristic matching. *arXiv preprint arXiv:1512.08422* .

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

Cicero D Santos and Bianca Zadrozny. 2014. Learning character-level representations for part-of-speech tagging. In *Proceedings of the 31st International Conference on Machine Learning (ICML-14)*. pages 1818–1826.

Milan Straka, Jan Hajic, and Jana Straková. 2016. Ud-pipe: Trainable pipeline for processing conll-u files performing tokenization, morphological analysis, pos tagging and parsing. In *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC 2016)*.

Kai Sheng Tai, Richard Socher, and Christopher D Manning. 2015. Improved semantic representations from tree-structured long short-term memory networks. *arXiv preprint arXiv:1503.00075* .

Adina Williams, Nikita Nangia, and Samuel R Bowman. 2017. A broad-coverage challenge corpus for sentence understanding through inference. *arXiv preprint arXiv:1704.05426* .