iUBC at SemEval-2016 Task 2: RNNs and LSTMs for interpretable STS

Iñigo Lopez-Gazpio and **Eneko Agirre** and **Montse Maritxalar**
IXA NLP group. University of the Basque Country (UPV/EHU)
Manuel Lardizabal 1, 20.018, Donostia, Basque Country
inigo.lopez@ehu.eus

Abstract

This paper describes *iUBC*, a neural network based approach that achieves competitive results on the interpretable STS task (iSTS 2016). Actually, it achieves top performance in one of the three datasets. iUBC makes use of a jointly trained classifier and regressor, and both models work on top of a recurrent neural network. Through the paper we provide detailed description of the approach, as well as the results obtained in iSTS 2015 test, iSTS 2016 training and iSTS 2016 test.

1 Introduction

Semantic Textual Similarity (STS) aims to catch the degree of equivalence between a pair of text nuggets. *Interpretable STS* (iSTS) is beyond STS in that it adds fine-grained information when evaluating the equivalence between text snippets. This explanatory layer is achieved by **aligning text segments** pertaining to one sentence with the segments pertaining to the second sentence, and, for each alignment, indicating a **relation label** and a **similarity score**.

In sum, alignments consist of a similarity score and a relation label that are defined as follows. On the one hand, the relation label has to be chosen from a set of categorical values (*equivalence, opposition, specialization, similarity and other kind of relation*). On the other hand, the similarity score has to be a real number bounded by (0,5]. Apart from this, there is an extra label to handle *not aligned* text segments.

The present paper describes *iUBC* and its participation in the *International Workshop on Semantic Evaluation* (SemEval-2016) task 2: Interpretable Semantic Textual Similarity. To check the task in full detail please refer to Agirre et al. (2016). Note that some of the authors participated in the organization of the task. Organizers prevented developers from access to the test dataset, and only allowed to access the same data as the rest of participants.

The paper is organized as follows. Section 2 describes iUBC's components, section 3 describes development performance and run configurations, section 4 shows the results obtained in the iSTS 2016 task, and, finally, section 5 mentions the conclusions and future work directions.

2 System Description

iUBC is composed of **three components**. The first component, *Input handling and chunking* (section 2.1), is responsible for reading the input and identifying segments over sentences; the second component, *Alignment* (section 2.2), takes care of aligning segments; and, finally, the third component, *Joint classification and scoring* (section 2.3), handles the assignment of similarity scores and relation labels. The main contribution of this architecture resides in the third component, in which a classifier and a regressor have been **trained jointly** on top of a recurrent artificial neural network (*A*NN).

2.1 Input handling and chunking

The iSTS task offers *two different scenarios* as regards the input: the scenario known as *System chunks* (syschunks), where participant systems are responsible for identifying the segments contained in the raw sentence pairs; and the *Gold chunks* scenario (goldchunks), where participants are provided

Proceedings of SemEval-2016, pages 771–776,
San Diego, California, June 16-17, 2016.

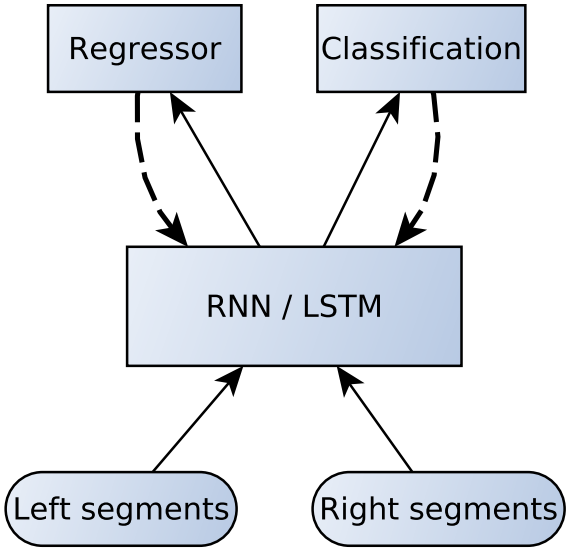

Figure 1: Joint model diagram. In the forward propagation (normal arrow) left and right segments are processed by a recurrent ANN producing as output a d-dimensional vector for each input segment. Features computed out of these vectors are then fed to both a regressor and a classifier that produce the similarity score and the relation label. In the backward propagation (stripped arrow) weights are adjusted in the recurrent ANN combining the gradients that propagate from the above models.

with gold standard segment marks over raw sentence pairs. The current component is only used in the syschunks scenario.

To identify and segment raw input sentences we use *python's NLTK library* (Bird, 2006) and the *ixa-pipes-chunker* (Agerri et al., 2014). Once the chunks are marked we use regular expressions to tune them according to the task's chunk definition. We developed four rules to optimize how conjunctions, punctuations and prepositions are handled. These rules aim to merge consequent chunks to form new chunks[1].

The output of the component are the same sentence pairs as the ones provided as input, but incorporating chunk marks to denote the start and end of segments.

[1]We found significant improvement if prepositional phrases followed by a nominal phrase are unified as a single chunk. The other three rules unify nominal phrases separated by punctuations, conjunctions, or a combination of them. The four rules are coded as regular expressions in Python.

2.2 Alignment

The alignment component focuses on making optimal segment connections for each sentence pair. The algorithm is as follows.

To begin with, the module constructs a token-token matrix in which each element (i,j) determines that there exists a connection between token i and token j[2]. The token-token matrix is populated using the weighted sum of the following metrics: lowercased token overlap, stemmed or lemmatized token overlap, cosine similarity between Mikolov's pretrained word vectors (Mikolov et al., 2013) and the alignment prediction provided by the monolingual word aligner described in Sultan et al. (2014).

Once the token-token matrix is built, the alignment component makes use of segment regions to group individual tokens. The strength of each segment connection is proportional to the weights of the interconnected tokens. By carrying out this operation over all segments in the pair the module obtains the chunk-chunk matrix[3]. Once the chunk-chunk matrix has been computed, the last step is to use the Hungarian-Munkres algorithm (Clapper, 2009) to discover the segments (x,y) that maximize the connection weights.

The alignment is done as follows: the segments that maximize the alignment strength are taken as *alignment main nodes*. Once the process is finished, the segments that are connected with lower weights to either one or the other of the main nodes are incorporated as *satellite nodes*. No many-to-many alignments are produced as we considered further analysis is necessary in order to obtain significant improvement.

2.3 Joint classification and scoring

iUBC uses a classification model and a regressor to predict the relation label and the similarity score for aligned segments.

The main picture of this component can be described as a **two layer architecture**, in which, a

[2]Token i being a token from the first sentence and token j being a token from the second sentence. Thus, the token-token matrix has a dimensionality of (#sentence1tokens x #sentence2tokens).

[3]This operation can be seen as pooling the token-token matrix by collapsing weights. The chunk-chunk matrix has a dimensionality of (#sentence1segments x #sentence2segments).

classifier and a regressor **work on top of a recurrent ANN**. While the models on the top layer are trained to produce scores and labels, the underlying recurrent net tries to capture the semantic representation of input segments and feed it upwards.

Both models on the top layer are trained in a supervised manner at the same time, and the delta error messages computed on them are used to train the net of the bottom layer. That is, **the gradient propagating from both models on the top layer** is used to train the weights of the ANN (Figure 1). A similar architecture with one top layer propagating gradients to an ANN has previously been used in Tai et al. (2015), which we use as motivation for our work.

The whole model works as follows: the ANN from the bottom layer processes segment words one at a time until there are no more words left. At each time step the net updates its *internal memory state* so that it keeps on capturing *the semantic representation of the segment*. Once the two segments have been processed the net outputs both segment representation d-dimensional vectors. These vectors are used to compute features for the models in the top layer.

Element wise *distance* ($|\vec{S1}^{\,d} - \vec{S2}^{\,d}|$) and *angle* ($\vec{S1}^{\,d} * \vec{S2}^{\,d}$) are computed as features, as proposed in Tai et al. (2015). The distance and angle concatenation yields a 2 * d-dimensional vector. This resulting vector is used as input in top layer models.

As regards the top layer models, feedforward neural networks are used for both. All the parameters of the models are summarized in Table 1. The scientific computing framework *Torch* has been used to build the whole component (Collobert et al., 2011). Note that this component doesn't use any type of lexicalized or domain specific feature but Pennington et al. (2014) word embeddings.

Bottom layer ANN: RNN or LSTM	
Input	Glove word embeddings
Output	Sentence representation
Input-dim	300
Memory-dim	150
Output-dim	150
Non linearity	Sigmoid function
Learning rate	0.05
Regularization	1e-4
Top layer - Regressor	
Input	Distance and angle
Output	Similarity score
Input-dim	2 * 150
Hidden-dim	50
Output-dim	1
Non linearity	Sigmoid function
Loss function	MSE criterion
Learning rate	0.05
Regularization	1e-4
Top layer - Classification model	
Input	Distance and angle
Output	Softmax pr among labels
Input-dim	2*150
Hidden-dim	50
Output-dim	5 (OPPO is not learnt)
Non linearity	Sigmoid function
Loss function	Multi-Class margin loss
Learning rate	0.05
Regularization	1e-4

Table 1: All parameters used in the Joint classification and scoring component.

3 Development

Initial experiments (section 3.2) have been carried out using the official train and test splits from iSTS 2015 (Agirrea et al., 2015). The 2016 interpretable STS task released three train datasets: *Images*, *Headlines* and *Answer-Students*. These datasets have been used to train the models using 10-fold cross-validation.

In Section 3.1 we describe in detail the set up of iUBC for each run, Section 3.2 presents the results on iSTS 2015 data, and in Section 3.3 we present the training results from iSTS 2016.

3.1 Selection of runs

We developed three runs with the following specific settings: **iUBC_run1**, the simplest run, uses a 1-layer RNN trained separately on each dataset; **iUBC_run2**, is the same as run1 but instead of using a 1-layer RNN it employs a 1-layer LSTM; **iUBC_run3**, is the same as run2 but the datasets are perturbed so they include segments that are not part of the gold standard. To produce this perturbation or noise we combine the gold standard alignments with the system alignments. The aim of doing so is to incorporate some noise in the training data, which we think would be useful to prevent overfitting. Both ANN models (RNN and LSTM) are coded following the equations of Tai et al. (2015).

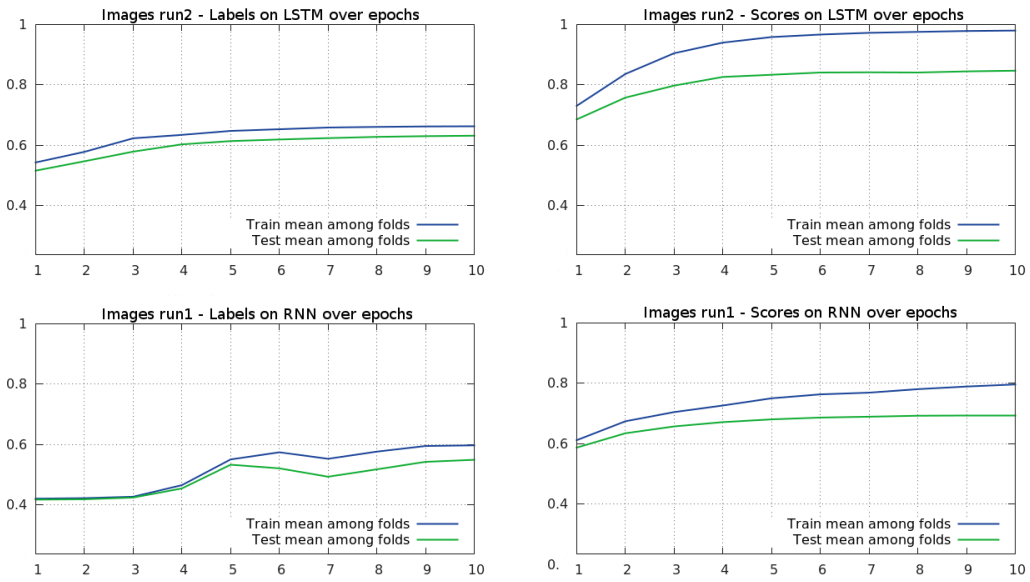

Figure 2: Cross-validation results for the *Joint classification and scoring* component over the iSTS 2015 training data. Figure describes run1 (RNN) and run2 (LSTM) results on the Images dataset. Micro F-score is used to evaluate labels and Pearson correlation coefficient is used to evaluate scores.

iSTS15	**H syschunks**				**H gschunks**			
	F	+T	+S	+TS	F	+T	+S	+TS
iUBC_r2	**0.78**	0.47	**0.71**	0.46	**0.91**	0.60	**0.83**	0.60
iUBC_r3	**0.78**	0.46	0.70	0.46	**0.91**	0.61	**0.83**	0.58
iUBC_r1	**0.78**	0.46	0.68	0.45	**0.91**	0.57	0.81	0.55
Baseline	0.67	0.46	0.60	0.46	0.84	0.56	0.76	0.56
AVG	0.69	0.45	0.61	0.43	0.84	0.56	0.75	0.54
MAX	**0.78**	**0.51**	0.70	**0.51**	0.90	**0.67**	**0.83**	**0.64**
	I syschunks				**I gschunks**			
	F	+T	+S	+TS	F	+T	+S	+TS
iUBC_r3	**0.85**	0.56	**0.78**	0.54	**0.90**	**0.61**	0.74	0.59
iUBC_r2	**0.85**	0.55	**0.78**	0.53	**0.90**	0.60	**0.84**	0.58
iUBC_r1	**0.85**	0.47	0.75	0.45	**0.90**	0.48	0.80	0.47
Baseline	0.71	0.37	0.61	0.37	0.84	0.43	0.72	0.43
AVG	0.67	0.41	0.59	0.39	0.82	0.50	0.72	0.47
MAX	0.83	**0.58**	0.75	**0.56**	0.89	**0.61**	0.80	**0.60**

Table 2: iSTS 2015 test results in Headlines and Images on both scenarios. Baseline, AVG and Max participants rows are taken from iSTS 2015. *F*, *+T*, *+S* and *+TS* stand for the official evaluation metrics F1 Alignment, F1 Type, F1 Score and F1 Type+Score.

3.2 Results on iSTS 2015 test

Results obtained using the described runs on iSTS 2015 test data are shown in Table 2. Comparing our runs to the published results, we think they perform competitively. According to the *F* evaluation metric, in both datasets we obtain equal or higher results than the maximum score among participants, moreover, our second run also obtains the highest results on the *+S* evaluation metric. The *+T* and *+TS* evaluation metrics are the ones in which our runs don't outperform best published results. Yet, they are above participants average in all scenarios, in some cases by large margin.

As regards our runs, we conclude that run2 and run3 outperform run1, but they both perform quite similarly. It seems that the noise added in the third run helps very slightly in the *Images* dataset. We also noticed that the hardest scenario for our runs turns to be the *Headlines syschunks*, where we almost obtain the same results for all runs.

As the majority of the systems participating in the iSTS 2015 task used lexicalized and task specific features or rules, we think iUBC is rather a different approach. It contributes to the task by being a system that doesn't make use of domain specific features but word embeddings while remaining competitive. We also think that results on the iSTS 2016 task will be reasonable as the training data for the iSTS 2016 task duplicates the one of the 2015. Actually, the reduced size of the training data is a matter that worried us. Due to this, for iSTS 2016 we decided not to divide the training data in train and development splits but to use cross-validation.

3.3 Results on iSTS 2016 train

Figure 2 shows cross-validation results for the *Joint classification and scoring* component over the *Images* dataset. Due to space constraint we have only included figures for the first and second run.

Comparing the RNN (Figure 2 bottom images) with the LSTM (Figure 2 top images) we can observe that the LSTM is able to fit the dataset with better results in fewer iterations. Actually, the evolution over epochs for the LSTM is smoother than the evolution of the RNN, especially for the labeling task. It seems that the RNN needs more epochs than the LSTM to fit the dataset.

It is also observable the high fitting of the LSTM to the scores of the training data, which is almost reaching the 100% correctness. Yet, this over-fitting seems not to contribute badly towards test results, which are noticeably higher than the RNN's. On the contrary, for relation labels the fitting is not that high for neither of the networks, even the LSTM outperforms the RNN.

4 Results on iSTS 2016 test

Table 3 shows the results obtained by distinct runs respectively in *Headlines*, *Images* and *Answer-Students* datasets.

Overall, iUBC performs competitively being *Headlines* the most difficult dataset to fit in and *Answer-Students* the best. In addition, we can see that both run2 and run3 outperform run1 by a large margin. Actually, the results scored by run1 are not that good as it scores below the participants' average. The main conclusion drawn from these result tables as regards run1 is that RNNs are not able to fit these datasets as well as LSTMs do.

Concerning run2 and run3 we expected run3 to outperform run2 on *syschunk* scenarios because of the noise it has been trained with. Nevertheless, this behavior can only be observed in *Headlines*, as in *images* run2 scores better than run3 and in *Answer-Students* they both score equally. The noise also seems not to affect *gschunks* scenarios very badly as in *Headlines* and *Anser-students* both runs score equally. Even though, run3's performance worsens 3 points in *Images* dataset on this scenario.

As pointed out in section 3.2 the *F* evaluation metric and the *+S* evaluation metric continue to be the ones in which iUBC scores best. Being sometimes (primarily on *gschunks* scenario) the top performing system above the participants' maximum. On the contrary, it is harder for the system to perform on the *+T* evaluation metric. This is related to what have been described in section 3.3. That is, the system finds it more difficult to fit the dataset labels than the scores. The main conclusion drawn from here could be that not only word embeddings, but also lexicalized features may be necessary in order to continue improving performance. Similar conclusions are achieved in Yin et al. (2015) regarding to the concatenation of word embedding based features and other kind of features.

iSTS16	**H syschunks**				**H gschunks**			
	F	+T	+S	+TS	F	+T	+S	+TS
iUBC_r3	0.81	0.51	0.74	0.50	**0.93**	0.60	**0.86**	0.59
iUBC_r2	0.81	0.49	0.74	0.48	**0.93**	0.60	**0.86**	0.59
iUBC_r1	0.81	0.43	0.71	0.42	**0.93**	0.51	0.83	0.50
Baseline	0.65	0.44	0.59	0.44	0.85	0.55	0.76	0.55
AVG	0.80	0.51	0.72	0.50	0.89	0.61	0.82	0.60
MAX	**0.84**	**0.56**	**0.76**	**0.55**	0.91	**0.70**	0.84	0.70
	I syschunks				**I gschunks**			
	F	+T	+S	+TS	F	+T	+S	+TS
iUBC_r2	**0.86**	0.56	**0.80**	0.55	**0.91**	0.62	**0.86**	0.61
iUBC_r3	**0.86**	0.52	**0.80**	0.52	**0.91**	0.59	0.85	0.58
iUBC_r1	**0.86**	0.49	0.77	0.48	**0.91**	0.52	0.82	0.51
Baseline	0.71	0.40	0.63	0.40	0.86	0.48	0.75	0.48
AVG	0.82	0.54	0.76	0.52	0.87	0.58	0.80	0.57
MAX	0.85	**0.63**	0.79	**0.61**	0.90	**0.69**	0.84	**0.67**
	AS syschunks				**AS gschunks**			
	F	+T	+S	+TS	F	+T	+S	+TS
iUBC_r3	0.80	**0.57**	0.75	**0.56**	**0.89**	**0.65**	**0.84**	**0.64**
iUBC_r2	0.80	**0.57**	0.75	**0.56**	**0.89**	**0.65**	**0.84**	**0.64**
iUBC_r1	0.80	0.45	0.71	0.45	**0.89**	0.50	0.79	0.50
Baseline	0.62	0.44	0.57	0.44	0.82	0.56	0.75	0.56
AVG	0.78	0.51	0.71	0.50	0.85	0.56	0.78	0.55
MAX	**0.82**	0.56	**0.76**	0.55	0.88	**0.65**	0.83	**0.64**

Table 3: iSTS 2016 test results in Headlines, Images and Answers-Students on both scenarios. Baseline, AVG and Max participants rows are taken from iSTS 2016. *F*, *+T*, *+S* and *+TS* stand for the official evaluation metrics F1 Alignment, F1 Type, F1 Score and F1 Type+Score.

5 Conclusions and Future Work

Throughout this paper we have described *iUBC*: a RNN and LSTM based system that has achieved reasonable results on the interpretable STS 2016 task. We have seen how the system works by **jointly training** a classifier and a regressor to produce the

relation labels and the similarity scores. We have also described how the error gradient from this top layer propagates to the bottom recurrent net, which aims to capture the semantic representation of input sentences into d-dimensional vectors.

We have shown performance of the system in the iSTS 2016 test data and described that iUBC **performs especially well** in the *Answer-Students dataset*. In addition, we have mentioned that the RNN based run is not able to perform as well as **LSTMs based runs**. Moreover, we have seen that including **noise in the training** data helps improve performance in the *Headlines* dataset on the *syschunks* scenario. But, worsens results in the *Images* dataset on the *gschunks* scenario.

We have also discussed that the model is more suitable to produce **similarity scores** than relation labels. This could be a consequence of the reduced size of the training data and labeling being a more demanding task. As regards this issue, we have mentioned that further features might be necessary in order to continue improving the system's results. In any case, this will require some more analysis.

All in all, we can conclude by saying that the interpretable STS task is an **interesting challenge** whose aim is to share knowledge about building NLP systems able to provide valuable feedback.

Acknowledgments

This material is based in part upon work supported a MINECO grant to the University of the Basque Country (TUNER project TIN2015-65308-C5-1-R). Iñigo Lopez-Gazpio is supported by a doctoral grant from MINECO. The IXA group is funded by the Basque Government (A type Research Group).

References

Rodrigo Agerri, Josu Bermudez, and German Rigau. 2014. IXA pipeline: Efficient and ready to use multilingual NLP tools. In *LREC*, volume 2014, pages 3823–3828.

Eneko Agirre, Aitor Gonzalez-Agirre, Iñigo Lopez-Gazpio, Montse Maritxalar, German Rigau, and Larraitz Uria. 2016. Semeval-2016 task 2: Interpretable semantic textual similarity. In *Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval 2016)*, San Diego, California, June.

Eneko Agirrea, Carmen Baneab, Claire Cardiec, Daniel Cerd, Mona Diabe, Aitor Gonzalez-Agirrea, Weiwei Guof, Inigo Lopez-Gazpioa, Montse Maritxalara, Rada Mihalceab, and others. 2015. Semeval-2015 task 2: Semantic textual similarity, english, spanish and pilot on interpretability. In *Proceedings of the 9th international workshop on semantic evaluation (SemEval 2015)*, pages 252–263.

Steven Bird. 2006. NLTK: the natural language toolkit. In *Proceedings of the COLING/ACL on Interactive presentation sessions*, pages 69–72. Association for Computational Linguistics.

BM Clapper. 2009. munkres: a python module implementing the hungarian method described by munkres (1957). version 1.0. 5.3.

Ronan Collobert, Koray Kavukcuoglu, and Clment Farabet. 2011. Torch7: A matlab-like environment for machine learning. In *BigLearn, NIPS Workshop*.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space.

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

Md Arafat Sultan, Steven Bethard, and Tamara Sumner. 2014. Back to basics for monolingual alignment: Exploiting word similarity and contextual evidence. 2:219–230.

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

Wenpeng Yin, Hinrich Schtze, Bing Xiang, and Bowen Zhou. 2015. ABCNN: Attention-based convolutional neural network for modeling sentence pairs.