Tree LSTMs for Learning Sentence Representations

Héctor Cerezo-Costas
AlantTic, Gradiant
Universidade de Vigo, Spain
Edificio CITEXVI, local 14
Vigo, Pontevedra 36310, SPA
hcerezo@gradiant.org

Manuela Martín-Vicente
Gradiant
Edificio CITEXVI, local 14
Vigo, Pontevedra 36310, SPA
mmartin@gradiant.org

F.J. González-Castaño
Dept. Enxeñaría Telemática
E.E. de Telecomunicación
Universidade de Vigo, SPA
javier@det.uvigo.es

Abstract

English. In this work we obtain sentence embeddings with a recursive model using dependency graphs as network structure, trained with dictionary definitions. We compare the performance of our recursive Tree-LSTMs against other deep learning models: a recurrent version which considers a sequential connection between sentence elements, and a bag of words model which does not consider word ordering at all. We compare the approaches in an unsupervised similarity task in which general purpose embeddings should help to distinguish related content.

Italiano. *In questo lavoro produciamo sentence embedding con un modello ricorsivo, utilizzando alberi di dipendenze come struttura di rete, addestrandoli su definizioni di dizionario. Confrontiamo le prestazioni dei nostri alberi-LSTM ricorsivi con altri modelli di apprendimento profondo: una rete ricorrente che considera una connessione sequenziale tra le parole della frase, e un modello bag-of-words, che non ne considera l'ordine. La valutazione dei modelli viene effettutata su un task di similarit non supervisionata, in cui embedding di uso generale aiutano a distinguere i contenuti correlati.*

1 Introduction

Word embeddings have succeeded in obtaining word semantics and projecting this information in a vector space. (Mikolov et al., 2013) proposed two methodologies for learning semantic abstractions of words from large volumes of unlabelled data, Skipgram and CBOW, comprised in the word2vec framework. Another approach is GloVe (Pennington et al., 2014), which learns from statistical co-occurrences of words. The two conceptually similar algorithms employ a sliding window of words, the context, with the intuition that words appearing frequently together are semantically related and thus should be represented closer in $\mathbb{R}^n$. The resulting vectors have shown strong correlation with human annotations in word-analogy tests (Griffiths et al., 2007).

Despite the success of word embeddings in capturing semantic information, they cannot obtain on its own the composition of longer constructions, which is essential for natural language understanding. Thus, several methods using deep neural networks combine word vectors for obtaining sentence representations with linear mappings (Baroni and Zamparelli, 2010) and deep neural networks, which make use of multiple network layers to obtain higher levels of abstraction (Socher et al., 2012). One of the first approaches of obtaining generic embeddings was Paragraph2Vec (Le and Mikolov, 2014). Paragraph2Vec can learn unsupervised sentence representations, analogous to word2vec models for word representation, by adding an extra node, indicating the document contribution, to the model.

Attending to the way the nodes of the network link with each other, two approaches are frequent in NLP: recurrent neural networks and recursive neural networks (RNN) [1]. Recurrent models consider sequential links among words, while recursive models use graph-like structures for organizing the network operations. They process neighbouring words by parsing the tree order (dependency or syntactic graphs), and compute node representations for each parent recursively from the previous step until they reach the root of the tree, which gives the final sentence abstraction.

In this work, we train a variant of Tree-LSTM models for learning concept abstractions with dic-

[1] We use the same classification as in (Li et al., 2015).

tionary descriptions as an input. To the best of our knowledge, this is the first attempt to embed dictionaries using such approach. Our model takes complex graph-like structures (e.g. syntactic or dependency graphs) as opposed to the most common approaches which employ recurrent models or unordered distributions of words as representation of the sentences. We use an unsupervised similarity benchmark with the intuition that better sentence embeddings will produce more coincidences with human annotations (comparably to the word analogy task in word embeddings).

2 Related Work

The following recurrent models are capable of obtaining general purpose embeddings of sentences: Skip-thought Vectors, and DictRep.

Skip-thought Vectors (Kiros et al., 2015) learns general semantic sentence abstractions with unsupervised training. This concept is similar to the learning of word embeddings with the skipgram model (Mikolov et al., 2013). Skip-thoughts tries to code a sentence in such a way that it maximises the probability of recovering the preceding and following sentence in a document.

DictRep (Hill et al., 2015) trains RNN networks and BoW models mapping definitions and words with different error functions (cosine similarity and ranking loss). Whilst the RNN models take into account the word orderings, the BoW models are just a weighted combination of the input embeddings. The simplest BoW approach offered competitive results against its RNN counterparts, beating them in most tests (Hill et al., 2016).

Recurrent models have achieved good performance results in different tasks such as polarity detection (e.g. bidirectional LSTMs in (Tai et al., 2015)), machine translation (Cho et al., 2014) or sentence similarity detection (e.g. Skip-thoughts), just to name a few.

Despite being less explored for building general purpose sentence embeddings, in several classification tasks, tree-structured RNNs represent the current state of the art. In their seminal paper, (Socher et al., 2013) captured complex interactions among words with tensor operations and graph-like links among network nodes. Recursive Neural Tensor Networks (RNTN) networks have been used to solve a simplified version of a QA system in (Iyyer et al., 2014).

In (Bowman, 2013), the authors built a natural language inference system using RNTN in a simplified scenario with basic sentence constructions. Although the results show that the system is able to learn inference relationships in most cases, it is unclear if this model could be generalised for more complex sentences. RNTNs were subsequently improved by (Tai et al., 2015), using LSTMs in the network nodes instead of tensors. With tree-structures the network can capture language constructions which greatly affect the polarity of sentences (e.g. negation, polarity reversal, etc.).

A more complete benchmark was conducted by (Li et al., 2015). There, sequential and recursive RNNs were tested in different tasks: sentiment analysis, question-answer matching, discourse parsing and semantic relation extraction. Recursive models excelled in tasks with enough available supervised data, when nodes different from the root are labelled, or when semantic relationships must be extracted from distant words in a sentence.

3 Approach

Learning models that build a dictionary of embeddings have solid advantages over other supervised approaches, since they take advantage of large volumes of data that are already available online. The training data of the system are pairs of definition/target word which can be built with dictionaries or encyclopedia descriptions (e.g. picking the first sentences of a description as training data). We follow previous work of (Hill et al., 2015) that employed dictionaries with sequential connections but using tree structures instead.

We used the Tree-LSTM as the starting point to build our system. The input to the system are the words conforming a definition together with the structure of the graph with the syntactic/dependency relationships, and the word closer to this definition, i.e. the target. Typically the LSTM nodes are intended for strictly sequential information propagation. Our variant is based in the previous work of (Tai et al., 2015).

The main differences with the original LSTM node are the presence of two forget gates instead of one and the operation over two previous nodes of the system which modify node states and inhibitor gates. Hence, sub-indexes 1 and 2 are reserved for left and right child nodes of the graph, respectively. In this LSTM node there are no peephole connections between memory states and the

inhibitor gates.

The state value in the root node is fed to the last layer of the system. Then, a non-linear transformation is applied to obtain the sentence embedding. In the basic configuration of the model, the error is measured by calculating the cosine similarity between target and predicted embeddings. The target is the embedding of the word result of the definition. Pre-trained word embeddings or random initialised embeddings might be employed. In the second case, the error is also propagated to the leaf nodes of the graph and thus the word embeddings are updated during training. We did not initialise randomly embeddings because this has consistently produced poorer results in comparison with the same model using pre-trained word embeddings.

In the network configurations of the tree-LSTM models, we added an extra backward link between the root node and the leaves reversing the uplink path (as hinted in (Socher et al., 2011; Paulus et al., 2014)). In these settings, the error to minimise is a combination of the target word similarity and the leaves word similarity modulated by a smoothing parameter.

We implemented our model with Theano (Theano Development Team, 2016) and trained it with minibatch (30) and Adam (Kingma and Ba, 2014) as optimisation algorithm (with parameters $\beta_1 = 0.9$, $\beta_2 = 0.999$ and learning rate $l = 0.002$). This configuration has achieved state of the art performance in other NLP tasks (Kumar et al., 2015).

4 Experiments

We compared DictRep (BoW and RNN) and our Tree-LSTM variant in a benchmark of unsupervised text similarity tasks and a supervised task (sentiment polarity). These tasks greatly benefit from a good representation of sentences and it requires a lot of human effort to build a dataset.

DictRep models were trained using available data and online code. For a fair comparison, all models employed the pre-trained word embeddings and training data provided by (Hill et al., 2015) and cosine similarity as error metric. The configuration setting was similar for all the models.

Our model employs two connection configurations: The Tree-LSTM with transformed dependency graphs and the sequential mapping of connections, which is conceptually similar to the DictRep-RNN model.

For SkipThoughts we used the code available online (ski,) and the pre-trained model with a sentence representation of 4800 dimensions. Additionally, we trained a compressed model with sentence and word representation dimensions of 1200 and 320 respectively in about three weeks. Like in the available model, the 80 million registers of the BookCorpus dataset (Zhu et al., 2015) were used during the training process.

The objective of the semantic similarity task benchmark is to measure the similarity between a pair of sentences. SemEval STS 2014 (Agirre et al., 2014) and SICK (Marelli et al., 2014) datasets were used for benchmarks. In both datasets, each example was gold-standard ranked between 0 (totally unrelated sentences) and 5 (completely similar). Furthermore, SICK dataset considers three different types of semantic relatedness (Neutral, Entailment and Contradiction). We tested the models against the three relations to check if recursive and recurrent models exhibited different behaviour.

This is the same dataset used in previous work (Hill et al., 2016) but excluding the WordNet set, since it was used as part of the training.

For the sentiment polarity, we used as training/validation data the Sentiment Penn Treebank dataset [2]. In this dataset, each sentence node is labelled with a 5-tag intensity tag from 0, the most negative, to 4. Sentences are already binarised in the same format of our TreeDict approach so that no preprocessing is needed in this task for TreeModels. We used for training and test the labels at the root node which is the the overall sentence polarity. For completeness, we repeat the analysis for a 3-label annotations over the same dataset. We used the same SVM classifier for all the models and we trained it with the sentence vectors as input.

5 Results and conclusion

The DictRep BoW model was undeniably better than the recurrent and recursive models achieving the best position in all cases (Table 1). The TreeDict-Dep model ranked second [3].

[2] http://nlp.stanford.edu/sentiment/treebank.html

[3] The character "-" indicates that some vectors for a sentence could not be obtained (e.g. due to a malformed dependency graph)

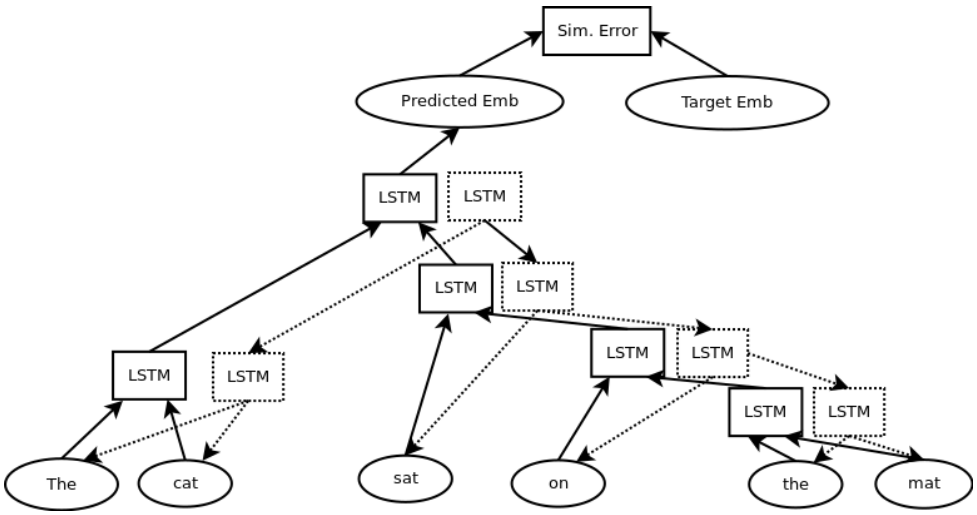

Figure 1: Tree-LSTM schema employed. Dotted blocks and lines depict the optional reverse channel.

All models capture the correlations with human annotations better in neutral contexts. If there are contradictions and entailment relationships, the agreement with human annotations is less evident. Nevertheless, this behaviour is expected and also desirable, as this is an unsupervised benchmark and the system has no way of learning a *similar but conflicting* relationship without external help.

It is clear that BoW models offered the best performance in all the datasets. The Tree-LSTM model, which is consistently better than the sequential models, ranked second. Table 2 shows the correlation among models over the SICK similarity dataset. All the models experience strong cross-correlations between them but the Tree-LSTM with dependency parsing showed the closest correlation with the BoW and recurrent models.

The Table 3 shows the performance of the models in the supervised polarity tasks. BoW and SkipThoughts models experience similar outcomes for the 5 and 3 label task. Models trained with dictionary definitions (DictRep and TreeDict) lag behind those models. However, all the networks using dependency structures have consistently beaten its sequential counterparts. This is a strong indicative of the benefits of using this more complex network structure. The difference between the different network configurations of the same model are less pronounced that in the similarity tasks but in our tests, the models that used the extra link backwards achieved small gains (at least in the 3-label task).

In previous work, (Hill et al., 2016) compared other models in this same similarity benchmark achieving comparable results. Not only DictRep-BoW models outperformed the DictRep-RNNs but also the Skip-thought model, which considers the order of the words in a sentence, was beaten by FastSent, its counterpart that employs BoW representation of a sentence.

The effect of word orderings is not clear. BoW models are far from being ideal as they cannot obtain which parts are negated or the dependencies among the different elements of the sentence (e.g. *the black dog chases the white cat* and *the black cat chases the white dog* cannot be differentiated by only using BoW models).

It is important to mention that the similarity was tested only at the root node when using Tree-LSTM. Notwithstanding, recursive models allow to use more elaborated strategies, taking advantage of the dependencies used to build the relationships of the nodes in the deep network. These strategies could combine similarities at different levels of the sentence to obtain a more approximate value of similarity (e.g. using a pooling matrix with all the nodes of the parse tree (Socher et al., 2011)).

The errors during training time in held-out data were 0.57 for BoW models versus the 0.51 achieved by recurrent and recursive models. Nevertheless, better dictionary embeddings do not seem to directly translate into better performance at inferring general purpose sentence embeddings in the benchmarks. Results in the test also show that we need better mechanisms to infer sentence level representations.

	STS 2014					Sick			
Model	News	Forum	Twitter	Images	Headlines	Neu	Ent	Con	All
DictRep-BoW	.67/.74	.42/.39	.60/.65	.71/.74	.58/.62	.60/.70	.58/.56	.12/.18	.62/.72
DictRep-RNN	.45/.52	.06/.04	.30/.32	.57/.57	.39/.42	.52/.59	.22/.23	.09/.10	.48/.56
TreeDict-Seq	.48/.54	.24/.23	.40/.45	.60/.64	.46/.51	.51/.59	.24/.27	.07/.10	.51/.59
TreeDict-Seq 250	.50/.58	.20/.21	.44/.47	.61/.66	.46/.49	.56/.62	.27/.30	.08/.11	.54/.64
TreeDict-Seq 250BL	.47/.47	.23/.21	.52/.59	.51/.51	.43/.45	.48/.52	.29/.33	.10/.14	.51/.56
TreeDict-Dep	.48/.55	.29/.28	-	.61/.67	-	.56/.64	.35/.39	.08/.13	.55/.65
TreeDict-Dep 250	.50/.56	.31/.30	-	.56/.63	-	.55/.61	.36/.41	.09/.12	.56/.63
TreeDict-Dep 250BL	.43/.45	.30/.28	-	.56/.58	-	.52/.56	.34/.38	.09/.11	.55/.60
SkipThoughts-4800	.43/.23	.13/.13	.42/.40	.48/.51	.36/.37	.49/.49	.19/.25	.10/.15	.48/.50
SkipThoughts-1200	.55/.54	.22/.23	-	.55/.61	.39/.41	.56/.56	.21/.24	.09/.15	.53/.56

Table 1: Performance of the models measured with Spearman/Pearson correlations against golden standard annotations in the similarity benchmarks.

Model	D.BoW	D.RNN	T.Seq	T.Penn
D.BoW	1.0/1.0	.70/.71	.74/75	.80/.82
D.RNN	.70/.71	1.0/1.0	.77/.75	.73/.72
T.Seq	.74/.75	.77/.75	1.0/1.0	.79/.78
T.Dep	.80/.82	.73/.72	.78/.78	1.0/1.0

Table 2: Spearman/Pearson correlations among the different models in the SICK dataset.

Model	F_1**-score**	
	(5-label)	**(3-label)**
DictRep-BoW	.40	.56
DictRep-RNN	.32	.49
TreeDict-Seq	.31	.49
TreeDict-Seq 250	.32	.48
TreeDict-Seq 250BL	.32	.49
TreeDict-Dep	.35	.53
TreeDict-Dep 250	.35	.51
TreeDict-Dep 250BL	.35	.53
SkipThoughts-4800	.40	.56
SkipThoughts-1200	.38	.55

Table 3: Performance of the models in the polarity detection task

In this paper we introduced the use of recursive models for the generation of general purpose embeddings once they are trained by embedding dictionary definitions. We compare recurrent and recursive models in the embedding dictionary task and we test the validity of these embeddings for their use as general purpose codification of sentences with both similarity.

Results demonstrate slight advantages of the Tree recursive variant over recurrent models that learn from dictionaries, which are more frequently employed. Recursive models are more expensive computationally and have a more complex implementation but they exhibit better performance in longer sentences. However, with current learning techniques recurrent and recursive models cannot offer better results than simpler models such as BoW representations of sentences in unsupervised similarity benchmarks. The results of these findings shall be confirmed in the future in more complex scenarios, such as large scale QA.

Acknowledgments

This work has been funded by the Spanish Ministerio de Economa y Competitividad through the project INRISCO (TEC2014-54335-C4-4-R).

References

Eneko Agirre, Carmen Banea, Claire Cardie, Daniel Cer, Mona Diab, Aitor Gonzalez-Agirre, Weiwei Guo, Rada Mihalcea, German Rigau, and Janyce Wiebe. 2014. Semeval-2014 task 10: Multilingual Semantic Textual Similarity. In *Proceedings of the 8th international workshop on semantic evaluation (SemEval 2014)*, pages 81–91.

Marco Baroni and Roberto Zamparelli. 2010. Nouns are vectors, adjectives are matrices: Representing adjective-noun constructions in semantic space. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, pages 1183–1193. Association for Computational Linguistics.

Samuel R Bowman. 2013. Can recursive neural tensor networks learn logical reasoning? *arXiv:1312.6192*.

Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning Phrase Representations using RNN Encoder-

decoder for Statistical Machine Translation. *arXiv preprint arXiv:1406.1078*.

Thomas L Griffiths, Mark Steyvers, and Joshua B Tenenbaum. 2007. Topics in Semantic Representation. *Psychological review*, 114(2):211.

Felix Hill, Kyunghyun Cho, Anna Korhonen, and Yoshua Bengio. 2015. Learning to Understand Phrases by Embedding the Dictionary. *Transactions of the Association for Computational Linguistics*.

Felix Hill, Kyunghyun Cho, and Anna Korhonen. 2016. Learning Distributed Representations of Sentences from Unlabelled Data. *arXiv:1602.03483*.

Mohit Iyyer, Jordan L Boyd-Graber, Leonardo Max Batista Claudino, Richard Socher, and Hal Daumé III. 2014. A Neural Network for Factoid Question Answering over Paragraphs. In *EMNLP*, pages 633–644.

Diederik Kingma and Jimmy Ba. 2014. Adam: A Method for Stochastic Optimization. *arXiv:1412.6980*.

Ryan Kiros, Yukun Zhu, Ruslan R Salakhutdinov, Richard Zemel, Raquel Urtasun, Antonio Torralba, and Sanja Fidler. 2015. Skip-Thought Vectors. In *Advances in neural information processing systems*, pages 3294–3302.

Ankit Kumar, Ozan Irsoy, Jonathan Su, James Bradbury, Robert English, Brian Pierce, Peter Ondruska, Ishaan Gulrajani, and Richard Socher. 2015. Ask Me Anything: Dynamic Memory Networks for Natural Language Processing. *arXiv preprint arXiv:1506.07285*.

Quoc V Le and Tomas Mikolov. 2014. Distributed Representations of Sentences and Documents. In *ICML*, volume 14, pages 1188–1196.

Jiwei Li, Minh-Thang Luong, Dan Jurafsky, and Eudard Hovy. 2015. When Are Tree Structures Necessary for Deep Learning of Representations? *arXiv:1503.00185*.

Marco Marelli, Stefano Menini, Marco Baroni, Luisa Bentivogli, Raffaella Bernardi, and Roberto Zamparelli. 2014. A SICK cure for the evaluation of compositional distributional semantic models. In *LREC*, pages 216–223.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient Estimation of Word Representations in Vector Space. *arXiv preprint arXiv:1301.3781*.

Romain Paulus, Richard Socher, and Christopher D Manning. 2014. Global belief recursive neural networks. In *Advances in Neural Information Processing Systems*, pages 2888–2896.

Jeffrey Pennington, Richard Socher, and Christopher D Manning. 2014. Glove: Global Vectors for Word Representation. In *EMNLP*, volume 14, pages 1532–43.

Sent2Vec encoder and training code from the paper "Skip-Thought Vectors". `https://github.com/ryankiros/skip-thoughts`. Accessed: 2017-07-07.

Richard Socher, Eric H Huang, Jeffrey Pennin, Christopher D Manning, and Andrew Y Ng. 2011. Dynamic Pooling and Unfolding Recursive Autoencoders for Paraphrase Detection. In *Advances in Neural Information Processing Systems*, pages 801–809.

Richard Socher, Brody Huval, Christopher D Manning, and Andrew Y Ng. 2012. Semantic Compositionality through Recursive Matrix-vector Spaces. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 1201–1211. Association for Computational Linguistics.

Richard Socher, Alex Perelygin, Jean Y Wu, Jason Chuang, Christopher D Manning, Andrew Y Ng, and Christopher Potts. 2013. Recursive Deep Models for Semantic Compositionality over a Sentiment Treebank. In *Proceedings of the conference on empirical methods in natural language processing (EMNLP)*, volume 1631, page 1642. Citeseer.

Kai Sheng Tai, Richard Socher, and Christopher D Manning. 2015. Improved Semantic Representations from Tree-structured Long Short-term Memory Networks. *ACL*.

Theano Development Team. 2016. Theano: A Python framework for fast computation of mathematical expressions. *arXiv e-prints*, abs/1605.02688, May.

Yukun Zhu, Ryan Kiros, Richard Zemel, Ruslan Salakhutdinov, Raquel Urtasun, Antonio Torralba, and Sanja Fidler. 2015. Aligning Books and Movies: Towards Story-like Visual Explanations by Watching Movies and Reading Books. In *arXiv preprint arXiv:1506.06724*.