Neural Classification of Linguistic Coherence using Long Short-Term Memories

Pashutan Modaresi
Institute of Computer Science
Heinrich Heine University of Düsseldorf
D-40225 Düsseldorf, Germany
modaresi@cs.uni-duesseldorf.de

Matthias Liebeck
Institute of Computer Science
Heinrich Heine University of Düsseldorf
D-40225 Düsseldorf, Germany
liebeck@cs.uni-duesseldorf.de

Stefan Conrad
Institute of Computer Science
Heinrich Heine University of Düsseldorf
D-40225 Düsseldorf, Germany
conrad@cs.uni-duesseldorf.de

ABSTRACT

Given a set of sentences, a sentence orderer permutes the sentences in a way that the final text is linguistically coherent and semantically understandable. In this work, we focus on the binary and ternary tasks of ordering a pair of sentences regarding their linguistic coherence. We propose a methodology to automatically collect and annotate sentence ordering corpora in the news domain for English and German documents. Furthermore, we introduce a data-driven end-to-end neural architecture to learn the order of a pair of sentences and also recognize the cases where no ordering can be determined due to missing context.

CCS Concepts

•Computing methodologies → Artificial intelligence; Natural language processing;

Keywords

Sentence ordering; long short-term memory; neural coherence classification

1. INTRODUCTION

The order of sentences in a document is what makes a text semantically meaningful. Assuming a single document $d = s_1, s_2, \ldots, s_n$, consisting of n sentences, there are $n!$ possible permutations of the sentences to form a document. However, despite this huge search space, humans are extraordinarily good at determining the order of sentences.

On the other hand, machines require the ability to deal with linguistic concepts such as *discourse coherence*, *linguistic redundancy and contradiction*, and, in general, *pragmatics* [3] to order sentences into a meaningful and coherent text.

Various approaches have been introduced to solve the problem of sentence ordering. In [19], a similarity metric is used to group the sentences into clusters and sentences are selected from clusters in a way to maximize the similarity between adjacent sentences. Lin et. al [9] make the assumption that a coherent text implicitly favors certain types of discourse relation transitions. The closest to our approach are the works of Lin and Jurafsky [8] and Chen et. al [1]. Using a large corpus of academic texts, Chen et. al train an algorithm to learn the pairwise ordering of sentences using various neural architectures. In a different approach, Lin and Jurafsky concatenate the sentences and train a classifier to decide whether the resulted text is coherent or not. We refer the reader to [10] for a detailed overview of the literature.

Despite having applications in fields such as text planning [7] and question-answering [16], multi-document summarization [12] is considered as one of the main applications of sentence ordering.

In this work, we define sentence ordering as a classification task realized by a function $\Phi : \mathbb{R}^m \times \mathbb{R}^{m'} \to \mathbb{Z}$, where $\mathbb{R}^m$ and $\mathbb{R}^{m'}$ are the corresponding vector representations of the input sentences in an arbitrary semantic space. Without loss of generality, we set $m = m'$ and $\mathbb{Z} = \{0, 1\}$ (binary classification) or $\mathbb{Z} = \{-1, 0, 1\}$ (multiclass classification). Given a permutation $\sigma \in \Sigma$ of a list of sentences $s = [s_1, \ldots, s_n]$, the optimal order σ^* can be computed as:

$$\sigma^* = \operatorname*{argmax}_{\sigma \in \Sigma} \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} \Phi(s_{\sigma(i)}, s_{\sigma(j)}) \tag{1}$$

Computing the optimal order using Equation 1 is computationally expensive. The focus of this work is to learn the function Φ and not to predict the optimal order σ^*. A possible strategy to compute σ^* is to use beam-search [1].

The open source implementation of our approach is hosted on Github[1].

2. ARCHITECTURE

As already stated, we treat the sentence ordering problem as a classification task. For this, we use a deep neural architecture to minimize the cross-entropy loss function [4].

$$\hat{p}(z|x_1; x_2) = \operatorname*{argmin}_{p(z|x_1;x_2)} \left\{ -\sum_{n \in N} \log p(z_n | x_1^{(n)}; x_2^{(n)}) \right\} \tag{2}$$

In Equation 2, N is the total number of training samples

[1]https://github.com/pasmod/reorderer

FIRE '16, December 08 - 10, 2016, Kolkata, India
© 2016 Copyright held by the owner/author(s). Publication rights licensed to ACM. ISBN 978-1-4503-4838-6/16/12...$15.00
DOI: http://dx.doi.org/10.1145/3015157.3015163

and $p(z_n|x_1^{(n)}; x_2^{(n)})$ is the estimate of the class probability of the n-th ordered pair $(x_1^{(n)}, x_2^{(n)})$ returned by the neural network.

The architecture of the network is depicted in Figure 1. The network has two inputs corresponding to the ordered sentence pair (s_1, s_2). We use a one-hot encoding to map each sentence s_i into a list of token indices in the vocabulary V and obtain its corresponding vector representation $\vec{s_i} \in \mathbb{R}^{|V|}$. Furthermore, we pad each vector with a special padding symbol to the maximum length of sentences in the corpus. Additionally, using an embedding layer, the one-hot encoded inputs are projected into a low-dimensional space. The embedding layer is realized by a simple matrix multiplication $\vec{e_i} = E \cdot \vec{s_i}$ with $E \in \mathbb{R}^{d \times |V|}$. The number of rows in the embedding matrix is set to $d = 200$ and it has as many columns as the size of the vocabulary.

We also initialize the matrix E with weights from pre-trained embeddings. For English, we use 200-dimensional embeddings trained using the GloVe [15] algorithm and for German we use 200-dimensional embeddings trained on German Wikipedia[2] using the continuous bag-of-words approach introduced in [11]. Initializing the embeddings matrix with pre-trained embeddings is especially advantageous when the size of the training data is limited.

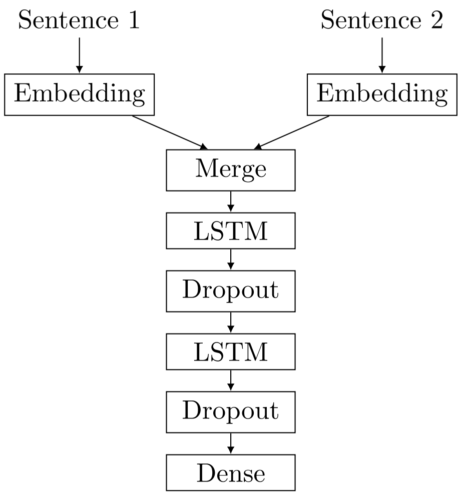

Figure 1: Deep learning architecture

The embeddings are then concatenated into a single vector $e = e_1 \oplus e_2$ with $e \in \mathbb{R}^{2d}$ that forms the input to a long short-term memory (LSTM) [6], which is a special kind of a recurrent neural network (RNN) addressing the difficulties in training RNNs [14].

In general, the hidden state of a vanilla RNN at time step t is updated as shown in Equation 3:

$$h_t = \sigma(W_{hh} h_{t-1} + W_{xh} x_t + b_h) \qquad (3)$$

In Equation 3, W_{hh} represents the recurrent weights from the hidden layer to itself, W_{xh} denotes the weights from the inputs to the hidden layer, b_h is the bias vector, and σ represents a non-linear function.

In LSTMs, the problem of long dependencies is addressed by the introduction of cell states and gates that regulate if information can be added or removed from cell state. In total, LSTMs consist of 3 gates (input, forget, and output) and a cell state, where the forget gate controls what information shall be thrown away, the input gate decides what information shall be stored in the cell state, and the output gate determines what information will be returned by the hidden state.

We use dropout as regularization technique [5] to avoid overfitting. The dropout operator simply sets a random subset of its arguments to zero. Following the work of Zaremba et. al [17], we apply the dropout operator only to the non-recurrent units.

In our architecture, we use two LSTMs and finally, feed a softmax layer with the hidden state of the second LSTM to transform the hidden state representation into predictive probabilities. This can be formalized as follows:

$$p_t = \text{softmax}(W_{hx} h_t + bx) \qquad (4)$$

We use Keras [2] to implement the proposed neural architecture. For the first LSTM we set the output dimension to 128 and for the second one to 64. Furthermore, we drop 50% of the non-recurrent units of the LSTMs. And finally to minimize the loss function (see Equation 2), we use an adaptive learning rate gradient descent method called Adadelta [18].

3. EXPERIMENTS

To train our neural architecture for the binary classification task, we used 30992 English and 22450 Germany samples and tested the final model with 7748 English and 5614 German unseen samples. For the ternary case, we used 50400 English and 36504 German samples for training and 12600 English and 9126 German samples for testing. Both train and test sets were balanced regarding the existing labels. In the following, the details of dataset construction and evaluation results are discussed in detail.

3.1 Dataset

To construct the underlying dataset for sentence ordering, we used the *Simurg*[3] [13] corpus which is an extendable multilingual collection of online news. In total, we used 9038 German and 12145 English news documents to train and validate our models.

To automatically label the dataset, we use a simple strategy. In the case of binary sentence ordering, for each ordered sentence pair (s_1, s_2), we define $\mathcal{L}(s_1, s_2) = 1$ if $s_1 \prec s_2$. Otherwise, we define $\mathcal{L}(s_1, s_2) = 0$. For this, we extract the first two sentences s_1 and s_2 of each news document, keep the natural order of the sentences and label them as positive. Additionally, we exchange the order of sentences and create a new ordered pair (s_2, s_1) with $\mathcal{L}(s_2, s_1) = 0$ to create negative examples.

The process for the ternary case is almost identical. We define $\mathcal{L}(s_1, s_2) = 1$ if $s_1 \prec s_2$ and $\mathcal{L}(s_1, s_2) = 0$ if $s_2 \prec s_1$. Furthermore, $\mathcal{L}(s_1, s_2) = -1$ if $s_1 \prec \ldots \prec s_i \prec \ldots \prec s_2$ or $s_2 \prec \ldots \prec s_i \prec \ldots \prec s_1$. For this, we extract the first and last sentence of each document and construct an ordered pair (s_1, s_2). This represents a situation where s_1 and s_2 are not adjacent and the ordered pair (s_1, s_2) is non-coherent due to missing context. In the remainder of this work we abbreviate this case with NC.

[2]https://de.wikipedia.org

[3]https://github.com/pasmod/simurg

3.2 Results

In total, we prepared an annotated dataset containing 63000 English and 42630 German samples for the ternary classification task and 38740 English and 28064 German samples for the binary classification problem. We randomly split the data into an 80% training set and a 20% validation set. We repeat the experiments three times and report the average number of false positives, false negatives, true positives, and true negatives together with their corresponding standard deviation.

Table 1: Confusion Matrix for English (Binary)

		Predicted		
		True	False	$\sum$
Actual	True	3703 ± 22	158 ± 22	3862
	False	223 ± 36	3662 ± 36	3886
	$\sum$	3927 ± 58	3820 ± 58	7748

The confusion matrices for the binary and ternary classification of English sentences are reported in Tables 1 and 2, respectively. As the experiments are conducted multiple times on various test sets, we report the mean values together with their corresponding standard deviation. Note that in the ternary case NC stands for "non-coherent".

Table 2: Confusion Matrix for English (Ternary)

		Predicted			
		True	NC	False	$\sum$
Actual	True	4132 ± 20	62 ± 18	5 ± 2	4200
	NC	77 ± 10	4153 ± 7	11 ± 3	4243
	False	45 ± 18	18 ± 14	4093 ± 13	4157
	$\sum$	4255 ± 44	4234 ± 28	4110 ± 18	12600

In both binary and ternary tasks, the test set is approximately balanced. It is also observable that the number of true positives and true negatives are very close to each other, which indicates that the classifier performs roughly equal for the existing labels.

Table 3: Confusion Matrix for German (Binary)

		Predicted		
		True	False	$\sum$
Actual	True	2654 ± 7	142 ± 7	2796
	False	151 ± 18	2667 ± 18	2818
	$\sum$	2805 ± 24	2809 ± 24	5614

The same properties also hold for the confusion matrices for the classification of German sentences. It is also observable that the standard deviations of diagonal elements are very low which indicates that the introduced neural architecture has a low variance. Also comparing the ratio of true negatives to the total number of samples yields that the classifier has a low bias. Notice that for readability we rounded all results in the confusion matrices to their next integers.

Additionally, we report the macro-averaged F_1 scores for both languages in Table 5. In the case of English sentences, the overall macro-averaged F_1 scores for the binary and the ternary classification tasks are 0.95 and 0.98, respectively. Interestingly, despite having an additional label in the ternary task, the F_1 score is higher than in the binary case.

Table 4: Confusion Matrix for German (Ternary)

		Predicted			
		True	NC	False	$\sum$
Actual	True	3065 ± 21	39 ± 16	12 ± 8	3117
	NC	43 ± 6	2974 ± 15	12 ± 10	3030
	False	38 ± 11	45 ± 15	2895 ± 5	2979
	$\sum$	3147 ± 38	3058 ± 46	2920 ± 9	9126

Table 5: Macro-Averaged F_1 Scores

	English		German	
	Binary	Ternary	Binary	Ternary
True	0.95	0.97	0.94	0.97
MC	—	0.98	—	0.97
False	0.95	0.98	0.94	0.98
Overall	0.95	0.98	0.94	0.97

The F_1 scores for the ternary classification of German and English sentences are almost identical and no significant difference could be observed. The same holds for the binary classification task.

Table 6: Macro-Averaged F_1 Scores (SVM)

	English		German	
	Binary	Ternary	Binary	Ternary
True	0.35	0.14	0.43	0.32
MC	—	0.14	—	0.12
False	0.12	0.20	0.07	0.04
Overall	0.24	0.16	0.25	0.16

We also compared the performance of our proposed algorithm to a baseline support vector machine (SVM) approach using the bag-of-words model. The results can be observed in Table 6 and can be compared to our results presented in Table 5. As expected, the SVM approach has much lower F_1 scores compared to our approach. Furthermore, no significant difference in F_1 scores for German and English can be observed. In general, support vector machines are not a suitable learning algorithm to model sequential data and thus have a poor performance on our data set.

4. CONCLUSIONS

We presented a neural architecture for classifying the linguistic coherence of a pair of sentences in German or English. In the binary case, we defined two sentences to be either adjacent or not and achieved an adequate F_1 score of 0.95 for English and 0.94 for German. In the ternary classification task, we defined a third label to represent the case when the sentences are non-coherent due to the missing context. For this task, we achieved an F_1 score of 0.98 for English and 0.97 for German sentences.

For future work, we plan to extend our corpora to assure the generalizability of our models and also increase the number of the inputs in the neural architecture to find the optimal order of multiple sentences without the use of search algorithms.

5. ACKNOWLEDGMENTS

This work was partially funded by the German Federal Ministry of Economics and Technology under the ZIM program (Grant No. KF2846504) and the PhD program *Online Participation*, supported by the North Rhine-Westphalian funding scheme *Fortschrittskollegs*. Computational support and infrastructure were provided by the "Centre for Information and Media Technology" (ZIM) at the University of Düsseldorf (Germany).

6. REFERENCES

[1] X. Chen, X. Qiu, and X. Huang. Neural Sentence Ordering. *ArXiv e-prints*, July 2016.

[2] F. Chollet. Keras. https://github.com/fchollet/keras, 2015.

[3] V. Fromkin, R. Rodman, and N. Hyams. *An Introduction to Language*. Cengage Learning, 2010.

[4] P. Golik, P. Doetsch, and H. Ney. Cross-Entropy vs. Squared Error Training: a Theoretical and Experimental Comparison. In *Interspeech*, pages 1756–1760, Aug. 2013.

[5] G. E. Hinton, N. Srivastava, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov. Improving neural networks by preventing co-adaptation of feature detectors. *CoRR*, abs/1207.0580, 2012.

[6] S. Hochreiter and J. Schmidhuber. Long Short-Term Memory. *Neural Comput.*, 9(8):1735–1780, Nov. 1997.

[7] E. H. Hovy. Planning Coherent Multisentential Text. In *Proceedings of the 26th Annual Meeting on Association for Computational Linguistics*, ACL '88, pages 163–169. Association for Computational Linguistics, 1988.

[8] J. Li and D. Jurafsky. Neural Net Models for Open-Domain Discourse Coherence. *CoRR*, abs/1606.01545, 2016.

[9] Z. Lin, H. T. Ng, and M.-Y. Kan. Automatically Evaluating Text Coherence Using Discourse Relations. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies - Volume 1*, HLT '11, pages 997–1006. Association for Computational Linguistics, 2011.

[10] W. Liu, X. Luo, J. Xuan, Z. Xu, and D. Jiang. Cognitive Memory-inspired Sentence Ordering Model. *Knowledge-Based Systems*, 104:1 – 13, 2016.

[11] T. Mikolov, K. Chen, G. Corrado, and J. Dean. Efficient Estimation of Word Representations in Vector Space. *CoRR*, abs/1301.3781, 2013.

[12] P. Modaresi and S. Conrad. On Definition of Automatic Text Summarization. In *Proceedings of Second International Conference on Digital Information Processing, Data Mining, and Wireless Communications*, DIPDMWC2015, pages 33–40. SDIWC, 2015.

[13] P. Modaresi and S. Conrad. Simurg: An Extendable Multilingual Corpus for Abstractive Single Document Summarization. In *Proceedings of the 8th Forum for Information Retrieval Evaluation*, FIRE '16. ACM, 2016.

[14] R. Pascanu, T. Mikolov, and Y. Bengio. On the Difficulty of Training Recurrent Neural Networks. *CoRR*, abs/1211.5063, 2012.

[15] J. Pennington, R. Socher, and C. D. Manning. GloVe: Global Vectors for Word Representation. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543, 2014.

[16] S. Verberne, L. Boves, N. Oostdijk, and P.-A. Coppen. Evaluating Discourse-based Answer Extraction for Why-question Answering. In *Proceedings of the 30th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '07, pages 735–736. ACM, 2007.

[17] W. Zaremba, I. Sutskever, and O. Vinyals. Recurrent Neural Network Regularization. *CoRR*, abs/1409.2329, 2014.

[18] M. D. Zeiler. ADADELTA: An Adaptive Learning Rate Method. *CoRR*, abs/1212.5701, 2012.

[19] R. Zhang. Sentence Ordering Driven by Local and Global Coherence for Summary Generation. In *Proceedings of the ACL 2011 Student Session*, HLT-SS '11, pages 6–11. Association for Computational Linguistics, 2011.