A Neural Local Coherence Model for Text Quality Assessment

Mohsen Mesgar *
Heidelberg Institute for
Theoretical Studies (HITS) and
Research Training Group AIPHES

Michael Strube
Heidelberg Institute for
Theoretical Studies (HITS)
michael.strube@h-its.org

Abstract

We propose a local coherence model that captures the flow of what semantically connects adjacent sentences in a text. We represent the semantics of a sentence by a vector and capture its state at each word of the sentence. We model what relates two adjacent sentences based on the two most similar semantic states, each of which is in one of the sentences. We encode the perceived coherence of a text by a vector, which represents patterns of changes in salient information that relates adjacent sentences. Our experiments demonstrate that our approach is beneficial for two downstream tasks: Readability assessment, in which our model achieves new state-of-the-art results; and essay scoring, in which the combination of our coherence vectors and other task-dependent features significantly improves the performance of a strong essay scorer.

1 Introduction

Coherence is a key factor that distinguishes well-written texts from random collections of sentences. A potential application of coherence models is text quality assessment. Examples include readability assessment (Pitler and Nenkova, 2008; Li and Hovy, 2014) and essay scoring (Miltsakaki and Kukich, 2004; Burstein et al., 2010). Here, we address the problem of local coherence modeling, which captures text relatedness at the level of sentence-to-sentence transitions.

Several approaches to local coherence modeling have been proposed. Entity-based methods principally relate adjacent sentences by means of entities, which are mentioned as noun phrases, NPs, in sentences (Barzilay and Lapata, 2008; Elsner and Charniak, 2011; Guinaudeau and Strube, 2013; Tien Nguyen and Joty, 2017). Lexical models connect sentences based on semantic relations between words in sentences (Beigman Klebanov and Shamir, 2006; Heilman et al., 2007; Mesgar and Strube, 2016). Both of these approaches suffer from different weaknesses. The entity-based models require an entity detection system, a coreference model, and a syntactic parser. These subsystems need to be perfect to gain the best performance of entity-based coherence models. The weakness of the lexical models is that they consider words independently, i.e. regardless of context in that words appear. More concretely, such lexical models take sentences as a bag of words. Recent deep learning coherence work (Li and Hovy, 2014; Li and Jurafsky, 2017) adopts recursive and recurrent neural networks for computing semantic vectors for sentences. Coherence models that use recursive neural networks suffer from a severe dependence on external resources, e.g. a syntactic parser to construct their recursion structure. Coherence models that purely rely on the recurrent neural networks process words sequentially within a text. However, in such models, long-distance dependencies between words cannot be captured effectively due to the limits of the memorization capability of recurrent networks.

Our motivation is to overcome these limitations. We use the advantages of distributional representations in order to, first, identify and represent salient semantic information that connects sentences, and second, extract patterns of changes in such information as a text progresses. By representing words of sentences with their pre-trained embeddings, we take lexical semantic relations between words into account. We employ a Recurrent Neural Network (RNN) layer to combine information in word embeddings and actual context information of words in sentences. Our model encodes salient information that relates two adja-

*This author is currently employed by the Ubiquitous Knowledge Processing (UKP) Lab, Technische Universität Darmstadt, https://www.ukp.tu-darmstadt.de.

cent sentences based on the two most similar RNN states in sentences. We accumulate two identified RNN states to represent semantic information that connects two adjacent sentences. We encode pattern of semantic information changes across sentences in a text by a convolutional neural network to represent coherence. Our end-to-end coherence model is superior to previous work because it relates sentences based on two semantic information states in sentences that are highly similar. So it does not need extra tools such as coreference resolution systems. Furthermore, our model incorporates words in their sentence context and models (roughly) distant relations between words.

We evaluate our model on two tasks: readability assessment and essay scoring. Both have been frequently used for coherence evaluation (Barzilay and Lapata, 2008; Miltsakaki and Kukich, 2004). Readability assessment is a ranking task where we compare the rankings given by the model against human judgments. Essay scoring is a regression task, in which we investigate if the combination of coherence vectors produced by our model and other essay scoring features proposed by Phandi et al. (2015) improves the performance of the essay scorer. The experimental results show that our model achieves the state-of-the-art result for readability assessment on the examined dataset (De Clercq and Hoste, 2016); and the combination of our coherence features with other essay scoring features significantly improves the performance of the examined essay scorer (Phandi et al., 2015).

2 Related Work

Early work on coherence captures different types of relations: entity-based (Grosz et al., 1995; Barzilay and Lapata, 2008), lexical-based (Beigman Klebanov and Flor, 2013; Somasundaran et al., 2014; Zhang et al., 2015), etc. Among these models, the entity-grid model (Barzilay and Lapata, 2005, 2008) has received a lot of attention. In this model, entities are defined, heuristically, by applying a string match over head nouns of all NPs in a text. The model, then, defines all possible changes over syntactic roles of entities in adjacent sentences as coherence patterns. The entity-grid model has been extended both by expanding its entity extraction phase (Elsner and Charniak, 2011; Feng and Hirst, 2012) and by defining other types of patterns (Lin et al., 2011; Louis and Nenkova, 2012; Ji and Eisenstein, 2014; Guinaudeau and Strube, 2013). Recently, Tien Nguyen and Joty (2017) fed entity grid representations of texts to a convolutional neural network (CNN) in order to overcome the limitation of predefined coherence patterns and extract patterns automatically. However, all of these models limit relations between sentences to entities that are shared by sentences. This makes the performance of these models dependent on the performance of other tools like coreference resolution systems and syntactic parsers. Our coherence model, in contrast, is based on relations between any embedded semantic information in sentences, and does not require entity annotations. A similar approach to ours is proposed by Mesgar and Strube (2016). Their approach encodes lexical relations between sentences in a text via a graph. Sentences are encoded by nodes, and lexical semantic relations between sentences are represented by edges. Coherence patterns are obtained by applying a subgraph mining method to graph representations of all texts in a corpus. This model involves words individually and independent of their sentence context. Our model uses a RNN layer over words in sentences to incorporate context information. Our approach for extracting coherence patterns also differs from this model as we employ CNNs rather than graph mining. Li and Hovy (2014) model sentences as vectors derived from RNNs and train a feed-forward neural network that takes an input window of sentence vectors and assigns a probability which represents the coherence of the sentences in the window. Text coherence is evaluated by sliding the window over sentences and aggregating their coherence probabilities. Similarly, Li and Jurafsky (2017) study the same model at a larger scale and use a sequence-to-sequence approach in which the model is trained to generate the next sentence given the current sentence and vice versa. Our approach differs from these methods; we represent coherence by a vector of coherence patterns. Moreover, our model takes distant relations between words in a text into account by relating two semantic states of sentences that are highly similar. Lai and Tetreault (2018) compare the performance of the aforementioned coherence models on texts from different domains. They conclude that the neural coherence models, which are explained above, surpass examined non-neural coherence models such as the entity-based models and the lexical-based model. Unlike their

evaluation method, which predicts the coherence level of a text, we rank two texts with respect to their coherence levels for the readability assessment task. We also show that integrating our coherence model into an essay scorer improves its performance.

An important task for evaluating a coherence model is readability assessment (Li and Hovy, 2014; Petersen et al., 2015; Todirascu et al., 2016). The more coherent a text, the faster to read and easier to understand it is. Early readability formulas were based on superficial text features such as average word lengths (Kincaid et al., 1975). These formulas systematically ignore many important factors that affect readability such as discourse coherence (Barzilay and Lapata, 2008). Schwarm and Ostendorf (2005) and Feng et al. (2010) recast readability assessment as a ranking task, and employ different semantic (e.g. language model perplexity scores) and syntactic (e.g. the average number of NPs) features to solve this task. Pitler and Nenkova (2008) show that discourse coherence features are more informative than other features for ranking texts with respect to their readability. Following the related work on coherence modeling (Barzilay and Lapata, 2008; Mesgar and Strube, 2015), we evaluate our coherence model on this task.

Another popular task for evaluating coherence models is essay scoring (Beigman Klebanov and Flor, 2013; Somasundaran et al., 2014). Miltsakaki and Kukich (2004) employ an essay scoring system to examine whether local coherence features, as defined by a measure of Centering Theory's Rough-Shift transitions (Grosz et al., 1995), might be a significant contributor to the evaluation of essays. They show that adding such features to their essay scorer improves its performance significantly. Burstein et al. (2010) specifically focus on the impact of entity transition features, as proposed by the entity-grid model for coherence modeling, on the essay scoring task. They demonstrate that by combining these features with other features related to grammar errors and word usage, the performance of their automated essay scoring system improves. Likewise, we combine our coherence vectors with other features that are used by a strong essay scorer (Phandi et al., 2015) and show that our coherence vectors improve the performance of this system significantly.

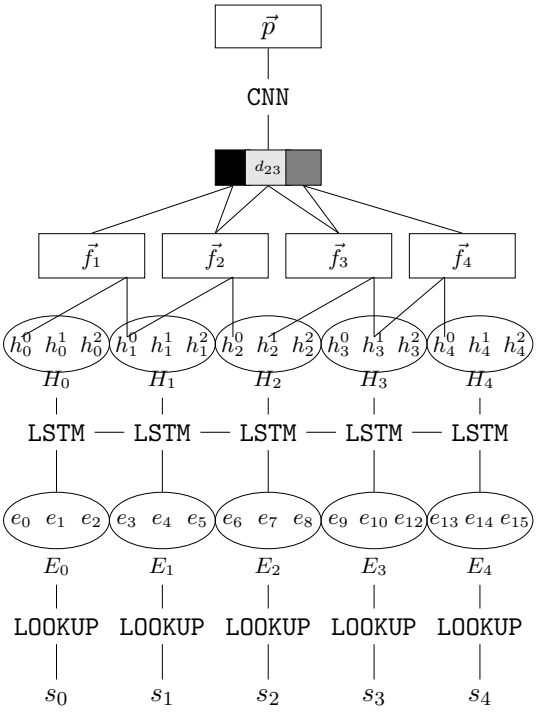

Figure 1: An illustration of our model. e_k is word embeddings associated with the k^{th} word in an input text. h_i^j depicts the j^{th} hidden state in LSTM states of sentence s_i. Two states in LSTM states of sentence s_i and sentence s_{i-1} that have the highest similarity are selected to connect sentences. Vector $\vec{f}_i$ captures information about the salient topic that relates sentence s_i to sentence s_{i-1}. d_{23} represents the similarity between $\vec{f}_2$ and $\vec{f}_3$ or the degree of continuity of the topic over adjacent sentences. Different shades of gray show different degrees of similarity. The CNN encodes patterns of changes as coherence vector $\vec{p}$.

3 Coherence Model

In this section, we describe details of our model. First, we explain how we encode words in their context (Section 3.1). Then we show how we relate sentences (Section 3.2), and finally we explain how we represent coherence based on sentence relations (Section 3.3). A general formulation of our model is a parametric function, $\vec{p} = L_\theta(d)$, where d is an input document, θ indicates parameters of neural modules, and $\vec{p}$ is a vector representation for the coherence of d. Figure 1 illustrates our model.

3.1 Word and Context Representations

We use a lookup table to associate all words in the vocabulary with word embeddings. The lookup table is initialized by existing pre-trained word embeddings because they capture lexical semantic re-

lations between words. For sentence s_i, the lookup table returns matrix E_i whose rows are embeddings of words in s_i. A weakness of former lexical coherence models (Somasundaran et al., 2014; Mesgar and Strube, 2016) is that they only rely on semantic relations between words in sentences, regardless of the current context of words. In order to overcome this limitation, we use a standard unidirectional[1] RNN with Long Short-Term Memory (LSTM) cells to encode the current context of words in sentences. For embedding matrix E_i:

$$H_i = \texttt{LSTM}\left(E_i, h_{i-1}^{n-1}\right),$$

where H_i is a list of LSTM states, and h_{i-1}^{n-1} is the last LSTM state of sentence s_{i-1}. Parameter n is the number of words in a sentence. We take state vector $h_i^j \in H_i$ as a representation of its input word embedding, e_j, that is combined with its preceding word vectors in sentence s_i. For sake of brevity, the details of LSTM formulations are explained in Appendix A.

3.2 Sentence Relation Representations

The relation between sentences is encoded by the most similar semantic states of sentences. Given two adjacent sentences, two of their LSTM states that have the highest similarity are selected to connect them. Those LSTM states refer to the salient semantic information that is shared between sentences. To model this, we follow attention components in neural language models (Bahdanau et al., 2014; Vaswani et al., 2017) where the similarity between the last LSTM state and each of its preceding states is computed to measure the amount of attention that the model should give to its preceding context for generating the next word. More formally, for two adjacent sentences s_i and s_{i-1}, one LSTM state in H_i and one LSTM state in H_{i-1} that have the maximum similarity are selected to represent the relation between the sentences:

$$(\vec{u}, \vec{v}) = \underset{\substack{(\vec{h}_m \in H_i) \\ (\vec{h}_n \in H_{i-1})}}{\text{argmax}} \; (sim(\vec{h}_m, \vec{h}_n)),$$

where H_i and H_{i-1} are LSTM states corresponding to sentences s_i and s_{i-1}. The similarity function, sim, returns the absolute value of the dot product between input vectors,

$$sim(\vec{h}_m, \vec{h}_n) = |\vec{h}_m \cdot \vec{h}_n|, \quad (1)$$

where the function $|.|$ computes the absolute value of its input[2]. We use the dot product function because in practice it enables our model to calculate the above equations efficiently in parallel and in matrix-space, i.e., directly on H_i and H_{i-1}. Since this is the details of implementation, we explain matrix-based equations in Appendix B. The absolute value in the similarity function is used to encode semantic relatedness between associated information with vectors, which is independent of the sign of the similarity function (Manning and Schütze, 1999).

We represent semantic information that relates two adjacent sentences by accumulating its selected LSTM states in the corresponding sentences. Since averaging in the vector space is an effective way to accumulate information represented in some vectors (Iyyer et al., 2015; Wieting et al., 2016), we compute the average of two identified vectors among the LSTM states of two adjacent sentences to represent semantic information shared by the sentences. More concretely, the vector representation of what relates sentence s_i to its immediately preceding sentence is obtained by averaging a vector of H_i and a vector of H_{i-1} that are identified as highly similar:

$$\vec{f}_i = avg(\vec{u}, \vec{v}) = \frac{\vec{u} + \vec{v}}{2},$$

where $\vec{u}$ and $\vec{v}$ are selected vectors. $\vec{f}_i$ is the vector representation of what connects s_i to its immediately preceding sentence.

3.3 Coherence Representations

Since sentences in a coherent text are about similar topics and share some semantic information, we compute semantic similarity between adjacent information states, i.e. $\vec{f}_i$s, to capture how they are changing through a text. We propose to encode changes by a continuous value between 0 and 1, where 1 shows that there is no change and 0 indicates that there is a big semantic drift in a text. Any value in between depicts how far a text is semantically changing. Given two adjacent vectors $\vec{f}_i$ and $\vec{f}_{i+1}$, the degree of continuity between them is:

[1] We use unidirectional RNN to model the way that an English text is read.

[2] In practice, the absolute function is implemented as $g(z) = \max(0, z) - \min(0, z)$ to be differentiable.

$$d = \frac{sim(\vec{f}_i, \vec{f}_{i+1})}{l},$$

where l is the length of input vectors, which is used to prevent large numbers (Vaswani et al., 2017), and sim is the similarity function (Section 3.2). The task of this layer is to check if the salient information that is shared by two adjacent sentences is salient in the subsequent sentence or not.

The last layer of our model is a convolutional layer to automatically extract and represent patterns of semantic changes in a text. CNNs have proven useful for various NLP tasks (Collobert et al., 2011; Kim, 2014; Kalchbrenner et al., 2014; Cheng and Lapata, 2016) because of their effectiveness in identifying patterns in their input (Xu et al., 2015). In the case of coherence, the convolution layer can identify coherence patterns that correlate with final tasks (Tien Nguyen and Joty, 2017). We use a temporal narrow convolution by applying a kernel filter k of width h to a window of h adjacent transitions over sentences to produce a new coherence feature. This filter is applied to each possible window of transitions in a text to produce a feature map $\vec{p}$, which is a coherence vector. Since we use a standard convolution layer, we explain details of the CNN formulations in Appendix C.

3.4 Variants of Our Model

In our experiments, we consider two variants of our model: **CohLSTM** that is the full version of our model as described above; and **CohEmb** that is an ablation. CohEmb has no RNN layer, so the model is built directly on word embeddings. In this model, relations between sentences are made over only content words by eliminating all stop words.

4 Implementation Details

Model configurations. Our model is implemented in PyTorch[3] with CUDA 8.0 support. In preprocessing we apply zero-padding to all sentences and documents to make their length equal. The vocabulary is limited to the 4000 most frequent words in the training data and all other words are replaced with the unknown token. We use the pre-trained word embeddings released by Zou et al. (2013), which are employed by state-of-the-art essay scoring systems. The dimensions of word embeddings and LSTM cells are 50 and 300, respectively. The convolution layer uses one filter with size 4. However, optimizing hyper-parameters for each task may lead to better performance. For selecting two vectors with the highest similarity from the LSTM states of two adjacent sentences, we capture the similarity between any pair of LSTM states of the sentences as an element in a vector, and then apply a max-pooling layer to this vector of similarities to identify the pair with maximally similar LSTM states. Selected LSTM states are used for representing salient information shared by the sentences. In CohEmb, stop words are removed by the SMART English stop word list (Salton, 1971).

Training setup. We set the mini-batch size to 32 and train the network for 100 epochs. At each epoch we evaluate the model on the validation set and select the one with the best performance for test evaluations. We optimize with *Adam*, with an initial learning rate of 0.01. Word vectors are updated during training. The dropout method with rate 0.5 is employed for regularization. Loss functions are specifically defined for each task.

5 Experiments

We evaluate our model on two downstream tasks: readability assessment (Section 5.1), in which coherence representations of documents are mapped to coherence scores, and then documents are ranked based on these scores; and essay scoring (Section 5.2), in which the coherence representation of an essay is combined with other features for essay scoring to quantify the quality of the essay.

5.1 Readability Assessment

Readability assessment – How difficult is a text to read and understand? – depends on many factors one of which is coherence. Texts that are more coherent are supposed to be faster to read and easier to understand. Following earlier research on local coherence (Barzilay and Lapata, 2008; Pitler and Nenkova, 2008; Guinaudeau and Strube, 2013), we evaluate our coherence model on this task by ranking texts with respect to readability, instead of predicting readability scores. More formally, we approach readability assessment as follows: Given a text-pair, which text is easier to read?

[3] https://pytorch.org

Compared models. We compare the two variants of our model as described in Section 3.4 with two following state-of-the-art systems:

Mesgar and Strube (2016). This is a graph-based coherence model, in which nodes of a graph indicate sentences of a text, and an edge between two sentence nodes represents the existence of a lexico-semantic relation between two words in the sentences. Semantic relations between words are measured by the absolute value of the cosine function over their corresponding pre-trained word embeddings. If the similarity value for two word vectors is below a certain threshold[4] then the connection between these two words is omitted. Given the graph representation of a text, its coherence is encoded as a vector whose elements are frequencies of different subgraphs in the graph. The size of subgraphs is defined by the number of their nodes and is set to five. Subgraphs are extracted by a random sampling approach. We choose this model for comparison because its intuition is similar to our model. However, this model suffers from the following limitations: word embeddings are considered independently, not in their current context; and a manual threshold is used for connection filtering. We overcome these two weaknesses using the RNN and CNN layers in our model, respectively.

De Clercq and Hoste (2016). This is the state-of-the-art readability system on the examined dataset. It uses a rich set of readability features ranging from surface to semantic text features. The ranking is performed by LibSVM in their model. We report their best performance that is achieved by extensive feature engineering and SVM's parameter optimization.

Experimental setup. In Section 3, we formulated our model as $\vec{p} = L_\theta(d)$ where θ represents parameters of the neural modules (i.e. the CNN and RNN layers) in our model. For this task, we use an output layer to map coherence vector $\vec{p}$ to score s which quantifies the degree of the perceived coherence of document d. Formally, the output layer is $s_d = \vec{u} \cdot \vec{p} + b$ where $\vec{u}$ and b are the weight vector and bias, respectively. Let document d be more readable than document d', then the model should ideally produce $s_d > s'_d$. We train the parameters of the model by a pairwise ranking approach and define the loss function as:

$$loss = \max\{0, 1 - s_d + s_{d'}\}.$$

The parameters of the model are shared to obtain the scores for texts in pair (d, d').

Data. We use the readability dataset proposed by De Clercq et al. (2014). It consists of 105 texts collected from the British National Corpus and Wikipedia in four different genres: administrative (e.g. reports and surveys), informative (e.g. articles of newspapers and Wikipedia entries), instructive (e.g. user manuals and guidelines), and miscellaneous (e.g., very technical texts and children's literature). The average number of sentences is about 12 per text. $10,907$ pairs of texts are labeled with five fine-grained categories: $\{-100, -50, 0, 50, 100\}$ indicating that the first text in a pair is respectively *much easier*, *somewhat easier*, *equally difficult*, *somewhat difficult*, *more difficult* to read than the second text in the pair. Labels of text-pairs are assigned by human judges. Similar to De Clercq and Hoste (2016), we evaluate on the positive and negative labels as two sets of classes resulting in $6,290$ text-pairs in total. The original readability dataset does not provide any standard training/validation/test sets. We apply 5-fold cross-validation over this dataset.

Evaluation metric. The quality of a model is measured in terms of accuracy, which is the fraction of pairs that are correctly ranked by a model divided by the total number of document-pairs. We report the average accuracy over all runs of cross-validation as the final result. We perform a paired t-test to determine if improvements are statistically significant ($p < .05$).

Results. Table 1 summarizes the results of different systems for the readability assessment task.

Model	Accuracy (%)
Mesgar and Strube (2016)	85.70*
CohEmb	92.17*
CohLSTM	**97.77**
De Clercq and Hoste (2016)	96.88

Table 1: Results on readability assessment. The first system is the state-of-the-art coherence model on this dataset. The last one is a full readability system. "$*$" indicates statistically significant difference with the bold result.

CohEmb significantly outperforms the graph-based coherence model proposed by

[4] Like Mesgar and Strube (2016), we set this threshold to 0.9.

Mesgar and Strube (2016) by a large margin (6%), showing that our model captures coherence better than their model. In our model, the CNN layer automatically learns which connections are important to be considered for coherence patterns, whereas this is performed in Mesgar and Strube (2016) by defining a threshold for eliminating connections.

CohLSTM significantly outperforms both the coherence model proposed by Mesgar and Strube (2016) and the CohEmb model by 11% and 5%, respectively, and defines a new state-of-the-art on this dataset. CohLSTM, unlike Mesgar and Strube (2016)'s model and CohEmb, considers words of sentences in their sentence context. This supports our intuition that actual context information of words contributes to the perceived coherence of texts.

CohLSTM, which captures exclusively local coherence, even outperforms the readability system proposed by De Clercq and Hoste (2016), which relies on a wide range of lexical, syntactic and semantic features.

5.2 Essay Scoring

One part of the student assessment process is essay writing where students are asked to write an essay about a given topic known as a prompt. An essay scoring system assigns an essay a score reflecting the quality of the essay. The quality of an essay depends on various factors including coherence. Following previous studies (Miltsakaki and Kukich, 2004; Lei et al., 2014; Somasundaran et al., 2014; Zesch et al., 2015), we approach this task by combining the coherence vector produced by our model and the feature vector developed by an open-source essay scorer to represent an essay by a vector. The final vector representation of an essay, $\vec{x}$, is mapped to a score by a simple neural regression method as follows:

$$s = sigmoid(\vec{u} \cdot \vec{x} + b),$$

where $\vec{u}$ and b are the weight vector and the bias, respectively. We exactly define vector $\vec{x}$ for different examined systems, where we explain compared models for essay scoring.

Compared models. We compare variations of our model (Section 3.4) with the following models:

EASE (BLRR). As a baseline we use an open-source essay scoring system, Enhanced AI Scoring Engine[5] (EASE) (Phandi et al., 2015). This system was ranked third among all 154 participating teams in the Automated Student Assessment Prize (ASAP) competition and is the best among all open-source participating systems. It employs Bayesian Linear Ridge Regression (BLRR) as its regression method applied to a set of linguistic features grouped in four categories: *(i) Frequency-based features*: such as the number of characters, the number of words, the number of commas, etc; *(ii) POS-based features*: the number of POS n-grams; *(iii) Word overlap with prompt*; *(iv) Bag of n-grams*: the number of uni-grams and bi-grams.

EASE. The difference between this system and *EASE (BLRR)* is in the employed regression method. This system uses a neural regression method as described above. In order to have a similar experimental settings for this task, here, we use feature vectors generated by Phandi et al. (2015) to train our neural regression system. The input of the neural regression function is a nonlinear transformation of the feature vector produced by EASE, $\vec{f}$. Therefore $\vec{x} = tanh(\vec{w} \cdot \vec{f} + b)$.

EASE & CohEmb. This model combines the feature vector computed by EASE, $\vec{f}$, and the coherence vector produced by CohEmb, $\vec{p}$, to have a more reliable representation of an essay. More concretely, the input to our regression function, $\vec{x}$, is obtained as follows:

$$\vec{h}_1 = tanh(\vec{w}_1 \cdot \vec{f} + b_1),$$
$$\vec{h}_2 = \vec{h_1} \oplus \vec{p},$$
$$\vec{x} = tanh(\vec{w}_2 \cdot \vec{h}_2 + b_2),$$

where $\oplus$ indicates the concatenation operatation.

EASE & CohLSTM. The structure of this model is the same as the *EASE & CohEmb* structure. But the input coherence vector, $\vec{p}$, is produced by CohLSTM.

Dong et al. (2017). It is a sentence-document model, which is especially designed for this task. It first encodes each sentence by a vector, which represents whole sentence meanings, and then use an RNN to embed vectors of sentences into a document vector.

Experimental setup. The size of the input vector for the regression method, $\vec{x}$, is fixed to 100 and its output size is fixed to 1. Dimensions of other parameters, $\vec{w}_1$ and $\vec{w}_2$, are set accordingly. The

[5] https://github.com/edx/ease

loss function is the Mean Squared Error (MSE) between human scores, $\vec{H}$, and scores predicted by our system, $\vec{S}$:

$$MSE(\vec{H}, \vec{S}) = \frac{1}{N} \sum (\vec{H} - \vec{S})^2.$$

The models are compared for each prompt by running 5-fold cross-validation (Dong et al., 2017).

Data. We apply our model to a dataset used in the Automated Student Assessment Prize (ASAP) competition run by Kaggle[6]. The essays are associated with scores given by humans and categorized in eight prompts. Each prompt can be interpreted as a different essay topic along with different genres. Table 2 summarizes some properties of this dataset.

Prompt	# Essays	Genre	Avg. Len.
1	1783	argumentative	350
2	1800	argumentative	350
3	1726	response	150
4	1772	response	150
5	1805	response	150
6	1800	response	150
7	1569	narrative	250
8	723	narrative	650

Table 2: Some properties of the dataset used for the essay scoring experiment.

Evaluation metric. ASAP adopted Quadratic Weighted Kappa (QWK) as the official evaluation metric. This metric measures the agreement between scores predicted by a system and scores assigned by humans. QWK considers chance agreements and penalizes large disagreements more than small agreements. We use an implementation of QWK that is described in Taghipour and Ng (2016). The formulation of QWK are explained in Appendix D. The final reported QWK is the average over QWKs of all prompts. We perform a paired t-test to determine if improvements are statistically significant ($p < .05$).

Results. Table 3 shows the results of different systems for the essay scoring task.

Both EASE & CohEmb, and EASE & CohLSTM significantly improve EASE, confirming that our proposed representation for coherence is beneficial for essay scoring and improves the performance of the examined essay scoring system. Our model does not beat the state-of-the-art essay scoring system (Dong et al., 2017), which is especially designed for this task and is tuned on this dataset. This model learns a vector representation for an input essay so that the vector performs the best for this regression task. In contrast, the core of our best performing essay scoring system, i.e. EASE & CohLSTM, is the feature vector generated by EASE, which has less modeling capacity than a deep learning model like the model proposed by Dong et al. (2017). The reason that we combine our coherence model with EASE, rather than the model proposed by Dong et al. (2017), is that EASE has no notion of coherence. By combining our coherence model with it, we examine if our coherence vector improves its performance or not.

Surprisingly, EASE & CohLSTM works on par with EASE & CohEmb. To gain a better insight, we ablate EASE feature vectors and compare the performance of the coherence models, i.e., CohLSTM, and CohEmb. Of course, coherence vectors on their own are not sufficient for predicting essay scores but this setup shows how much each variant of our model contributes to this task. The two last rows in Table 3 show the results.

CohLSTM outperforms CohEmb on all prompts, which matches the results for readability assessment. This confirms our intuition that integrating the information of the current context of words contributes to coherence measurement.

In terms of average QWK, CohLSTM works similar to EASE; however they behave differently on different prompts. The largest improvement for CohLSTM, with respect to EASE, is obtained on prompt 7 and 8. These two prompts ask for stories about *laughter* and *patience*, so corresponding essays can be categorized in the narrative genre (see Table 2). The guidelines of these two prompts, which are publicly available in the Kaggle data, ask human annotators to assign the highest score to essays that are coherent and hold the attention of readers through an essay. This is what our model captures: the sequence of semantic changes in a text, or coherence.

On prompt 5, in contrast, we see the largest deterioration in performance of CohLSTM in comparison to EASE. This prompt asks students to describe the mood created by the author of a memoir. Essays are expected to contain specific infor-

[6] https://www.kaggle.com/c/asap-aes/data

Model	Prompts								Avg QWK
	1	2	3	4	5	6	7	8	
EASE (BLRR)	0.761	0.606	0.621	0.742	0.784	0.775	0.730	0.617	<u>0.705</u>
Dong et al. (2017)	0.822	0.682	0.672	0.814	0.803	0.811	0.801	0.705	0.764
EASE	0.702	0.572	0.620	0.731	0.752	0.758	0.648	0.530	0.664
EASE & CohEmb	0.783	0.646	**0.664**	0.776	0.777	0.776	0.744	0.632	0.725*
EASE & CohLSTM	**0.784**	**0.654**	0.663	**0.788**	**0.793**	**0.794**	**0.756**	**0.646**	**0.728***
CohEmb	0.625	0.523	0.501	0.570	0.581	0.578	0.661	0.472	0.564
CohLSTM	0.669	0.634	0.591	0.710	0.639	0.716	0.729	0.641	0.666

Table 3: Results on essay scoring. "$*$" shows significant improvements with respect to the underlined score ($p < .05$). Bold numbers show the best results among different variants of our model.

mation from the memoir so that an essay with the highest score has the highest coverage of all relevant and specific information from the memoir. Therefore, mentioning the details of the memoir in essays of prompt 5 is more important than coherence for this prompt. This also shows that our model exclusively captures the coherence of a text, which is the goal of this paper.

6 Conclusions

We developed a local coherence model that encodes patterns of changes in what semantically relates adjacent sentences. The main novelty of our approach is that it defines sentence connections based on any semantic concept in sentences. In this sense, our model goes beyond entity-based coherence models, which need extra dependencies such as coreference resolution systems. Moreover, in contrast to lexical cohesion models, which take words individually, our model encodes words in their sentence context. Our model relates sentences by means of distant relations between word representations. The most similar LSTM states in two adjacent sentences are selected to encode the salient semantic concept that relates the sentences. The model finally employs a convolutional layer to extract and represent patterns of topic changes across sentences in a text as a coherence vector.

We evaluate coherence vectors generated by our model on the readability assessment and essay scoring tasks. On the former, our model achieves new state-of-the-art results. On the latter, it significantly improves the performance of a strong essay scorer. We believe the reason that our system works is that it learns which semantic concepts of sentences should be used to relate sentences, and which information about concepts is required to model sentence-to-sentence transitions. In future work we intend to run qualitative experiments on patterns that are extracted by our model to see if they are also linguistically interpretable.

Acknowledgments

This work has been supported by the German Research Foundation (DFG) as part of the Research Training Group Adaptive Preparation of Information from Heterogeneous Sources (AIPHES) under grant No. GRK 1994/1 and the Klaus Tschira Foundation, Heidelberg, Germany. We thank Mohammad Taher Pilehvar, Ines Rehbein, and Mark-Christoph Müller for their valuable feedback on earlier drafts of this paper. We also thank anonymous reviewers for their useful suggestions for improving the quality of the paper.

References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*.

Regina Barzilay and Mirella Lapata. 2005. Modeling local coherence: An entity-based approach. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics,* Ann Arbor, Mich., 25–30 June 2005, pages 141–148.

Regina Barzilay and Mirella Lapata. 2008. Modeling local coherence: An entity-based approach. *Computational Linguistics*, 34(1):1–34.

Beata Beigman Klebanov and Michael Flor. 2013. Word association profiles and their use for automated scoring of essays. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers),* Sofia, Bulgaria, 4–9 August 2013, pages 1148–1158.

Beata Beigman Klebanov and Eli Shamir. 2006. Reader-based exploration of lexical cohesion. *Language Resources and Evaluation*, 40(2):109–126.

Jill Burstein, Joel Tetreault, and Slava Andreyev. 2010. Using entity-based features to model coherence in student essays. In *Proceedings of Human Language Technologies 2010: The Conference of the North American Chapter of the Association for Computational Linguistics,* Los Angeles, Cal., 2–4 June 2010, pages 681–684.

Jianpeng Cheng and Mirella Lapata. 2016. Neural summarization by extracting sentences and words. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers),* Berlin, Germany, 7–12 August 2016, pages 484–494.

Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa. 2011. Natural language processing (almost) from scratch. *Journal of Machine Learning Research*, 12(8):2493–2537.

Orphée De Clercq and Véronique Hoste. 2016. All mixed up? Finding the optimal feature set for general readability prediction and its application to English and Dutch. *Computational Linguistics*, 42(3):457–490.

Orphée De Clercq, Véronique Hoste, Bart Desmet, Philip Van Oosten, Martine De Cock, and Lieve Macken. 2014. Using the crowd for readability prediction. *Natural Language Engineering*, 20(3):293–325.

Fei Dong, Yue Zhang, and Jie Yang. 2017. Attention-based recurrent convolutional neural network for automatic essay scoring. In *Proceedings of the 21st Conference on Computational Natural Language Learning,* Vancouver, Canada, 3–4 August 2017, pages 153–162.

Micha Elsner and Eugene Charniak. 2011. Extending the entity grid with entity-specific features. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers),* Portland, Oreg., 19–24 June 2011, pages 125–129.

Lijun Feng, Martin Jansche, Matt Huenerfauth, and Noémie Elhadad. 2010. A comparison of features for automatic readability assessment. In *Proceedings of Coling 2010: Poster Volume,* Beijing, China, 23–27 August 2010, pages 276–284.

Vanessa Wei Feng and Graeme Hirst. 2012. Extending the entity-based coherence model with multiple ranks. In *Proceedings of the 13th Conference of the European Chapter of the Association for Computational Linguistics,* Avignon, France, 23–27 April 2012, pages 315–324.

Barbara J. Grosz, Aravind K. Joshi, and Scott Weinstein. 1995. Centering: A framework for modeling the local coherence of discourse. *Computational Linguistics*, 21(2):203–225.

Camille Guinaudeau and Michael Strube. 2013. Graph-based local coherence modeling. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers),* Sofia, Bulgaria, 4–9 August 2013, pages 93–103.

Michael J. Heilman, Kevyn Collins-Thompson, Jamie Callan, and Maxine Eskenazi. 2007. Combining lexical and grammatical features to improve readability measures for first and second language texts. In *Proceedings of Human Language Technologies 2007: The Conference of the North American Chapter of the Association for Computational Linguistics,* Rochester, N.Y., 22–27 April 2007, pages 460–467.

Mohit Iyyer, Varun Manjunatha, Jordan Boyd-Graber, and Hal Daumé III. 2015. Deep unordered composition rivals syntactic methods for text classification. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers),* Beijing, China, 26–31 July 2015, pages 1681–1691.

Yangfeng Ji and Jacob Eisenstein. 2014. Representation learning for text-level discourse parsing. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers),* Baltimore, Md., 22–27 June 2014, pages 13–24.

Nal Kalchbrenner, Edward Grefenstette, and Phil Blunsom. 2014. A convolutional neural network for modelling sentences. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers),* Baltimore, Md., 22–27 June 2014, pages 655–665.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing,* Doha, Qatar, 25–29 October 2014, pages 1746–1751.

J. Peter Kincaid, Robert P. Jr. Fishburne, Richard L. Rogers, and Brad S. Chisson. 1975. Derivation of new readability formulas (automated readability index, Fog count and Flesch reading ease formula) for navy enlisted personnel. Technical Report 8-75, Naval Technical Training Command, Naval Air Station Memphis-Millington, Tenn.

Alice Lai and Joel Tetreault. 2018. Discourse coherence in the wild: A dataset, evaluation and methods. In *Proceedings of the SIGdial 2018 Conference: The 19th Annual Meeting of the Special Interest Group on Discourse and Dialogue,* Melbourne, Australia, 12–14 July 2018, pages 214–223.

Chi-Un Lei, Ka Lok Man, and To Ting. 2014. Using learning analytics to analyze writing skills of students: A case study in a technological common core curriculum course. *IAENG International Journal of Computer Science*, 41(3):41–45.

Jiwei Li and Eduard Hovy. 2014. A model of coherence based on distributed sentence representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing,* Doha, Qatar, 25–29 October 2014, pages 2039–2048.

Jiwei Li and Dan Jurafsky. 2017. Neural net models of open-domain discourse coherence. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing,* Copenhagen, Denmark, 7–11 September 2017, pages 198–209.

Ziheng Lin, Hwee Tou Ng, and Min-Yen Kan. 2011. Automatically evaluating text coherence using discourse relations. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers),* Portland, Oreg., 19–24 June 2011, pages 997–1006.

Annie Louis and Ani Nenkova. 2012. A coherence model based on syntactic patterns. In *Proceedings of the 2012 Conference on Empirical Methods in Natural Language Processing and Natural Language Learning,* Jeju Island, Korea, 12–14 July 2012, pages 1157–1168.

Christopher D. Manning and Hinrich Schütze. 1999. *Foundations of Statistical Natural Language Processing*. MIT Press, Cambridge, Mass.

Mohsen Mesgar and Michael Strube. 2015. Graph-based coherence modeling for assessing readability. In *Proceedings of STARSEM 2015: The Fourth Joint Conference on Lexical and Computational Semantics,* Denver, Col., 4–5 June 2015, pages 309–318.

Mohsen Mesgar and Michael Strube. 2016. Lexical coherence graph modeling using word embeddings. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies,* San Diego, Cal., 12–17 June 2016, pages 1414–1423.

Eleni Miltsakaki and Karen Kukich. 2004. Evaluation of text coherence for electronic essay scoring systems. *Natural Language Engineering*, 10(1):25–55.

Casper Petersen, Christina Lioma, Jakob Grue Simonsen, and Birger Larsen. 2015. Entropy and graph based modelling of document coherence using discourse entities: An application to IR. In *Proceedings of the ACM SIGIR International Conference on the Theory of Information Retrieval,* Northhampton, Mass., 27-30 September 2015, pages 191–200.

Peter Phandi, Kian Ming A. Chai, and Hwee Tou Ng. 2015. Flexible domain adaptation for automated essay scoring using correlated linear regression. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing,* Lisbon, Portugal, 17–21 September 2015, pages 431–439.

Emily Pitler and Ani Nenkova. 2008. Revisiting readability: A unified framework for predicting text quality. In *Proceedings of the 2008 Conference on Empirical Methods in Natural Language Processing,* Honolulu, Hawaii, 25–27 October 2008, pages 186–195.

Gerard Salton. 1971. *The SMART Retrieval System – Experiments in Automatic Document Processing*. Englewood Cliffs, N.J.: Prentice Hall.

Sarah E. Schwarm and Mari Ostendorf. 2005. Reading level assessment using support vector machines and statistical language models. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics,* Ann Arbor, Mich., 25–30 June 2005, pages 523–530.

Swapna Somasundaran, Jill Burstein, and Martin Chodorow. 2014. Lexical chaining for measuring discourse coherence quality in test-taker essays. In *Proceedings of COLING 2014, the 25th International Conference on Computational Linguistics: Technical Papers,* Dublin, Ireland, 23–29 August 2014, pages 950–961.

Kaveh Taghipour and Hwee Tou Ng. 2016. A neural approach to automated essay scoring. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing,* Austin, Tex., 1–5 November 2016, pages 1882–1891.

Dat Tien Nguyen and Shafiq Joty. 2017. A neural local coherence model. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers),* Vancouver, Canada, 30 July – 4 August, 2017, pages 1320–1330.

Amalia Todirascu, Thomas Francois, Delphine Bernhard, Nuria Gala, and Anne-Laure Ligozat. 2016. Are cohesive features relevant for text readability evaluation? In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers,* Osaka, Japan, 11–17 December 2016, pages 987–997.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *31st Conference on Neural Information Processing Systems (NIPS 2017),* Long Beach, CA., USA, 4–9 December 2017, pages 6000–6010.

John Wieting, Mohit Bansal, Kevin Gimpel, and Karen Livescu. 2016. Charagram: Embedding words and sentences via character n-grams. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing,* Austin, Tex., 1–5 November 2016, pages 1504–1515.

Kelvin Xu, Jimmy Ba, Ryan Kiros, Kyunghyun Cho, Aaron Courville, Ruslan Salakhudinov, Rich Zemel, and Yoshua Bengio. 2015. Show, attend and tell:

Neural image caption generation with visual attention. In *Proceedings of the 32nd International Conference on Machine Learning (Volume 37)* , Lille, France, 07–09 July 2015, pages 2048–2057.

Torsten Zesch, Michael Wojatzki, and Dirk Scholten-Akoun. 2015. Task-independent features for automated essay grading. In *Proceedings of the 10th Workshop on Innovative Use of NLP for Building Educational Applications,* Denver, Col., 4 June 2015, pages 224–232.

Muyu Zhang, Vanessa Wei Feng, Bing Qin, Graeme Hirst, Ting Liu, and Jingwen Huang. 2015. Encoding world knowledge in the evaluation of local coherence. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies,* Denver, Col., 31 May – 5 June 2015, pages 1087–1096.

Will Y. Zou, Richard Socher, Daniel Cer, and Christopher D. Manning. 2013. Bilingual word embeddings for phrase-based machine translation. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing,* Seattle, Wash., 18–21 October 2013, pages 1393–1398.

A LSTM Layer

The LSTM layer is defined as follows:

$$
\begin{aligned}
i_t &= \sigma(W_{ii}e_t + b_{ii} + W_{hi}h_{(t-1)} + b_{hi}),\\
f_t &= \sigma(W_{if}e_t + b_{if} + W_{hf}h_{(t-1)} + b_{hf}),\\
g_t &= \tanh(W_{ig}e_t + b_{ig} + W_{hc}h_{(t-1)} + b_{hg}),\\
o_t &= \sigma(W_{io}e_t + b_{io} + W_{ho}h_{(t-1)} + b_{ho}),\\
c_t &= f_t * c_{(t-1)} + i_t * g_t,\\
h_t &= o_t * \tanh(c_t),
\end{aligned}
$$

where e_t is the input word embeddings.

B Matrix-based Equations

Given the matrix representations of two adjacent sentences (e.g. H_i and H_j), the similarity matrix between LSTM states of these two sentences is defined as follows:

$$SIM = |H_i \cdot H_j^T|,$$

where SIM is the similarity matrix and H_j^T is the transpose of matrix H_j.

$$I_{index}, J_{index} = \texttt{ARGMAX}\left(SIM\right),$$

$$\vec{u}, \vec{v} = H_i[I_{index}], H_j[J_{index}],$$

where the ARGMAX function determines the indices (the row index I_{index} and the column index J_{index}) of the maximum element of the SIM matrix. These indices point out to word vectors of H_i and H_j that are considered for representing the sentence relation.

C Convolution Layer

A convolution operation involves applying filter w (i.e. a vector of weight parameters) to the vector of similarities of k continuity degrees among adjacent sentences in order to encode local transitions of the salient topic:

$$\vec{c} = \tanh(w^T \cdot L_{t:t+k-1} + b_t),$$

where $L_{t:t+k-1}$ denotes the k elements in the vector representation of degrees of continuity and b_t is the bias. Notice that we use a wide convolution, as opposed to narrow, to ensure that the filters reach entire elements of an input vector, including the boundaries. We do this by performing zero-padding, where elements located out of boundaries are assumed to be zero.

D QWK

To calculate QWK, between two sets of scores, a weight matrix W is constructed as follows:

$$W_{ij} = \frac{(i-j)^2}{(N-1)^2},$$

where i is the rating assigned by a human annotator and j is the rating assigned by a system. N is the number of possible ratings. A matrix O is calculated such that $O_{i,j}$ is the number of essays that receive a rating i by the human annotator and a rating j by the system. The last matrix is E that is calculated by the outer product of the histogram vectors of the human and system ratings. The matrix E is then normalized such that the sum of the elements in E and the sum of the elements in O are the same. QWK is calculated using the matrices W, O, and E as follows:

$$QWK = 1 - \frac{\sum_{i,j} W_{ij}O_{ij}}{\sum_{ij} W_{ij}E_{ij}}.$$