Text Coherence Analysis on Social Media

Aman Raj (006) **Tushar Bhutada (073)** **Udbhav Chugh (081)** **Mayank Baranwal (084)**
Indian Institute of Technology Guwahati
Group 8, End-Term Report
{aman170101006, tusha170101073, udbha170123055, baran170102035}@iitg.ac.in

Abstract

With social media becoming the number one platform for individuals to find a global audience, being coherent is more crucial than ever. However, the colloquialisms of social media make assessing coherence a problematic task. In this project, we look to automate the generation a dataset of coherent and incoherent social media posts. We also describe the preprocessing pipeline, a critical step given the varied nature of social media. Furthermore, we develop a sliding window convolutional neural network (CNN) architecture to capture the semantics of social media. Finally, we compare our results with existing architectures on both social media and non-social media-based corpora.

1 Introduction

Coherence is a field of linguistics that focuses on the characteristics of semantically meaningful text. Coherent passages have smooth transitions with different thoughts flowing evenly from one paragraph to the next. Ensuring coherence is key to enabling easy comprehension of an idea. A lack of coherence can convolute the most simple concept beyond comprehension.

1.1 Coherence in Social Media

Social media is a platform that provides all voices with a global reach; clear messaging has the potential to be heard across the world. Writing sensible sentences is not enough; an argument is perceived as coherent only if the ideas flow smoothly from one paragraph to another. More often than not, social media users start a discussion on a particular topic but finish on an entirely different theme. Worse still, malicious actors may choose to obscure the facts by convoluting a topic beyond comprehension. Thus, it is necessary for social media users to not only be coherent but also have the capability to recognise an incoherent argument.

To analyse and overcome these issues, we aimed to generate new datasets that reflect the conversational nature of social networking discussions. Furthermore, we looked at implementing a novel architecture for better assessment of user post coherency. We now discuss the existing literature in section 2, our methodology in sections 3, 4 and 5 and the results in section 6.

2 Related Work

2.1 Available Datasets

2 corpora are widely used for text coherence analysis(Barzilay and Lapata, 2005). The first is a collection of aviation accident reports from National Transportation Safety Board's (NTSB). The second contains Associated Press articles on the topic of earthquakes. Both the datasets contain ordered pairs of alternative renderings (x_{ij}, x_{ik}) of the same document d_i, where x_{ij} exhibits a higher degree of coherence than x_{ik}. Existing datasets are synthetically generated using an original document (x_{ij}), taking its different permutations and forming the pair (x_{ij}, x_{ik}) with x_{ij} being the original (and more coherent) document.

2.2 Text Coherence Models

A variety of approaches have been utilised to tackle the problem of text coherence. (Barzilay and Lapata, 2005) created an entity-grid representation of documents, that captures distributional, syntactic, and referential information patterns of entity distribution. (Elsner and Charniak, 2008) developed a discourse-new classifier and pronoun coreference method for the information ordering task in text coherence analysis. (Mesgar and Strube, 2016) implemented a lexical coherence graph to represent lexical relations among sentences. With this, sentence flow was described on semantic relations between words.

(Li and Jurafsky, 2017) used neural models to measure multiple aspects of coherence in existing and generated sentences. (Mesgar and Strube, 2018) used recurrent neural networks (RNN) and Long short-term memory (LSTM) for computing semantic vectors for sentences. Here, the information in word embeddings and the surrounding context is combined using RNNs. (Cui et al., 2017) used a CNN architecture to capture text coherence. They also calculated the similarities of their distributional representation and captured the interactions between sentences and analyse coherence.

2.3 Task Description

The principal goals of this project are two-fold. Firstly, we aim to generate a new dataset for training models for text coherence analysis in a social media context. Secondly, we aim to create a novel model architecture for coherence analysis on social media text and analyse its performance on text coherence applications in general. The details of our methodology are described in detail in the following sections.

3 Method Part 1: Datasets

3.1 Requirement of Separate Datasets

Social Media posts and texts vary significantly from other forms of texts. The tone is often informal and is often accompanied by heavy usage of slang and short forms which do not occur in regular corpora. Colloquialisms also make it hard to identify and classify keywords and entities. Furthermore, in many social media posts, users will start a discussion on a particular topic but finish on a tangential note. Finally, there is a considerable variation in themes and a lack of consistent structure in many posts. Hence, it is essential to have a corpus that reflects the nature of social media if we want the model to learn the appropriate characteristics.

3.2 Dataset Characteristics

Taking into consideration the feedback from our professors, we decided to generate two different datasets. Of these, one has the posts and comments as separate documents (as planned earlier). The other one had posts and comments concatenated to have a better flavour of discussion on a topic as suggested by sir.

Among the popular social media platforms, after analysing the posts and discussions on them, we chose Reddit Top 1000 posts dataset provided publicly on Kaggle because of its varied types of conversations and no character limit. This enabled us to create a dataset which can capture social media discussions of a wide variety of topics and post lengths. We chose 4 different subreddit topics, namely *Books, Explain like I'm five, AskReddit* and Jokes to have diverse dataset characteristics. We extracted an equal number of documents from each subreddit to keep a consistent yet broad dataset. The exact details are provided in subsection 3.3.

3.3 Dataset Extraction and Generation

We used Selenium tool to extract required parts from Reddit webpages to form the coherent part of our dataset. Similar to (Barzilay and Lapata, 2005), we generated pairs of documents with one document more coherent than the other. To achieve this, we labelled the original post as coherent and the permutations of its sentences as incoherent. The underlying assumption is that the original sentence order in the source document must be more coherent than its permutation. Since the coherence relation between different permutations is unknown, our corpus includes pairwise rankings that comprise the original document and one of its permutations.

For permutation generation, we first broke down each document into sentences, for which we used the $sent\_tokenize$ method of the NLTK tool. We took 70 documents from each subreddit having a minimum of 1000 characters each to ensure we have over 10-12 lines for each document to analyse text coherence appropriately. For each subreddit, we split the documents into 63 training and 7 testing documents. For the training document, we took the original post (indicating coherent document) along with one permutation (indicating incoherent document) to form 126 training points for each subreddit. Hence we generated a total of $4 * 126 = 504$ documents for training. For the 7 documents separated for testing per subreddit, we generated 20 permutations of the original document to generate $7 * 20 = 140$ document pairs (x_i, x_j) where x_i is more coherent than x_j. Hence we generated a total of $4 * 140 = 560$ documents for testing. This process is repeated for both the datasets: one in which the posts and comments are considered separate documents and another where posts and top 10 comments are considered together.

Algorithm 1 Dataset generation

linenosize=

Input: $\mathcal{R}D$ – Reddit Top 1000 posts dataset
Output: $\mathcal{D}$ – Dataset

```
 1: function GENERATEDATASET
 2:     $requiredDocs \leftarrow 70$
 3:     $limit \leftarrow 1000$
 4:     for $subreddit \in \mathcal{R}D$ do
 5:         $urls \leftarrow list(subreddit)$
 6:         $found \leftarrow 0$
 7:         for $url \in urls$ do
 8:             $post \leftarrow getPost(url)$
 9:             $comts \leftarrow getComments(url)$
10:             $doc \leftarrow post$
11:             for $comt \in comts[0:10]$ do
12:                 $doc \leftarrow doc + comt$
13:             end for
14:             if $charCount(doc) \geq limit$ then
15:                 $found \leftarrow found + 1$
16:                 $sents \leftarrow sentTokenize(doc)$
17:                 if $doc$ to be added in Test then
18:                     $permutCount \leftarrow 20$
19:                 else
20:                     $permutCount \leftarrow 1$
21:                 end if
22:                 for $k \leftarrow 1$ to $permtCount$ do
23:                     $new \leftarrow shuffle(sents)$
24:                     $\mathcal{D} \leftarrow \mathcal{D} + pair(doc, new)$
25:                 end for
26:             end if
27:             if $found = requiredDocs$ then
28:                 break
29:             end if
30:         end for
31:     end for
32:     return $\mathcal{D}$
33: end function
```

Algorithm 1 shows the steps for dataset generation of type 2 dataset where post and top 10 comments are concatenated together. For type 1 dataset, the only difference in its algorithm is that instead of concatenating posts and comments together, they are checked separately for character count greater than thousand and are added in dataset as separate documents. This approach of corpora generation for social media texts is similar to what has been used in standard text coherence analysis applications, as explained before.

4 Method Part 2: Preprocessing

Having extracted the dataset, we now need to refine it for model training. This step is especially critical for social media texts as they tend to use a large number of colloquialisms. We now detail the pipeline specified in Figure 1.

4.1 Cleaning up the Dataset

We created the following pipeline for data preprocessing:

1. **Sentence Level Cleaning**: The first step is to extract the sentences from each post. We do this with the aid of NLTK's $sent\_tokenize$ package. Then, we proceed to remove all non-alphanumeric characters from the text. The removal of these characters ensures that a model does not recognize the same entity as separate ones due to erroneous punctuation. It also allows us to remove nouns (such as emails, hashtags etc) that can serve as red-herrings for coherency evaluation.

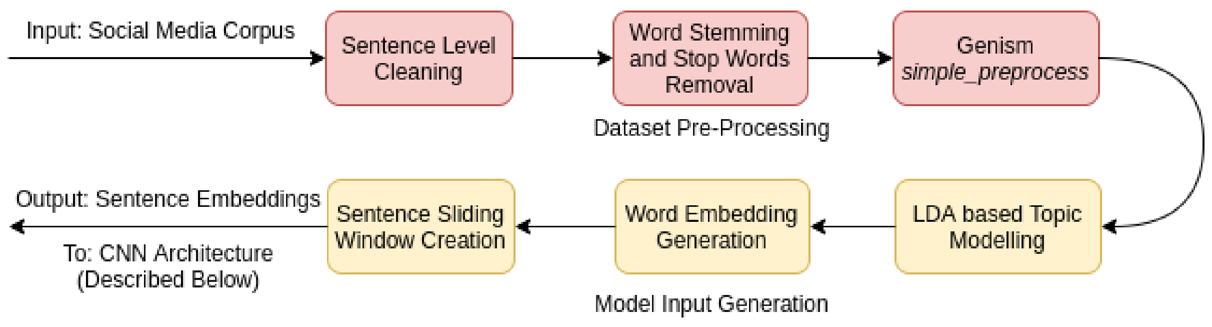
2. **Word/Token Level Cleaning**: We stem each word with NLTK's $PorterStemmer$ package and stores its root. Each word is also passed through a set of pre-defined filters to remove stop words. Finally, only words that fall under the category of noun, adjective, verb or adverb are included.

   In posts on social media, punctuation and grammar may not always be appropriately utilised. Stemming words and extracting nouns, adjectives etc allows a model to accommodate a variety of writing styles. It also ensures that during training, the model is not chasing a "moving goalpost" due to variance in writing formats.

Additionally, the data is also passed through genism's $simple\_preprocess$. This deals with more peculiar character level modifications like lowercasing, de-accentations etc. At the end of this stage, we have the relevant entities that can be used for inferring text coherency. However, multiple unrelated entities may be present that can mislead the model in detecting coherency.

4.2 Extracting Relevant Entities

We now counter off-topic entities with topic modelling through Latent Dirichlet Allocation (LDA)(Jelodar). With the aid of LDA, we generate some of the dominating themes in the text. Note that these themes are statistically developed and are thus arbitrary from a linguistic standpoint. Each topic has a weighted contribution from different words. We then select the entities that match these topics and use them for generating word embeddings. As we reorder the sentences to generate incoherent corpus pairs, we note that the entities extracted are same and it is their order that helps distinguish the incoherency.



Input: Social Media Corpus
Sentence Level Cleaning
Word Stemming and Stop Words Removal
Genism simple_preprocess
Dataset Pre-Processing
LDA based Topic Modelling
Word Embedding Generation
Sentence Sliding Window Creation
Output: Sentence Embeddings
To: CNN Architecture (Described Below)
Model Input Generation

Figure 1: Pipeline for Model Preprocessing

4.3 Generating Word Embeddings

To allow the model to make sense of the words, we generate their respective word-embeddings. For this purpose, we had initially employed gensim's partially trained *Word2Vec* model. Currently, we have replaced this with *GloVE* embeddings as they are completely pre-trained and provide similar results. We pass the pre-processed entities from our corpus and obtain their embeddings.

To aid in visualization, we show some of the resulting word embeddings, compressed to 2 dimensions in Figure 3. We observe that the entities of a similar topic (defined by legend specifying the dominant keyword) are typically clustered together, as expected.

5 Method Part 3: Text Coherence Architecture

In this section, we will discuss the architecture designed to use the generated word embeddings from section 4.3 for text coherence analysis.

5.1 Proposed Architecture

Now that we have generated the word embeddings for the entities, we use the following architecture to train our model. We have used similar idea to use CNN as taken up in (Cui et al., 2017):

- For every post in training data, we use a sliding window approach where the window size is equal to 3 sentences. This enables us to capture the context of nearby sentences in coherent vs non-coherent passages.

- For the extracted entities in each of these 3 sentences, we generate word embeddings from section 4.3 to form three 2D matrices S_i, S_{i+1}, and S_{i+2} of dimensions W_i x d where W_i is the number of entities extracted for sentence S_i using the process mentioned before, and d is the dimension of word embeddings which we have taken to be equal to 50.

- On each of the 3 matrices, we apply a convolutional layer with kernel $size = 3$ x d, number of filters(C) = 16, $stride = 1$ and a zero-padded row at beginning and end of the matrix. The convolution aids in capturing spatial locality of the word embeddings. This gives us 3 matrices with i^{th} matrix of dimensions C x W_i x 1.

- The matrices are then passed through a layer of average pooling resulting in 3 one dimensional vectors of size $C = 16$. This layer normalizes varying input sentence lengths to a fixed length.

- The 3 matrices are then flattened and concatenated to form a layer of size $3 * C = 48$. The next layer is a hidden neural network of the same dimension, $3 * C = 48$. The output of this hidden layer then passes through an average pooling layer to give a final tensor of dimension 1. We introduced the hidden layer to allow for greater parameter tuning.

- The sigmoid function is applied to this value to get the probability, which indicates the coherence score for the particular window. The coherence score for the post is calculated using the product of coherent scores of all windows of the post as mentioned in (Li and Hovy, 2014). We then back-propagate the loss (explained in section 5.2) to train the network. Overall, in a comparison between two documents, the document with higher coherence score will be considered more coherent than the other. Applying the sigmoid function provides us with a differentiable operator that provides the output between 0 and 1.

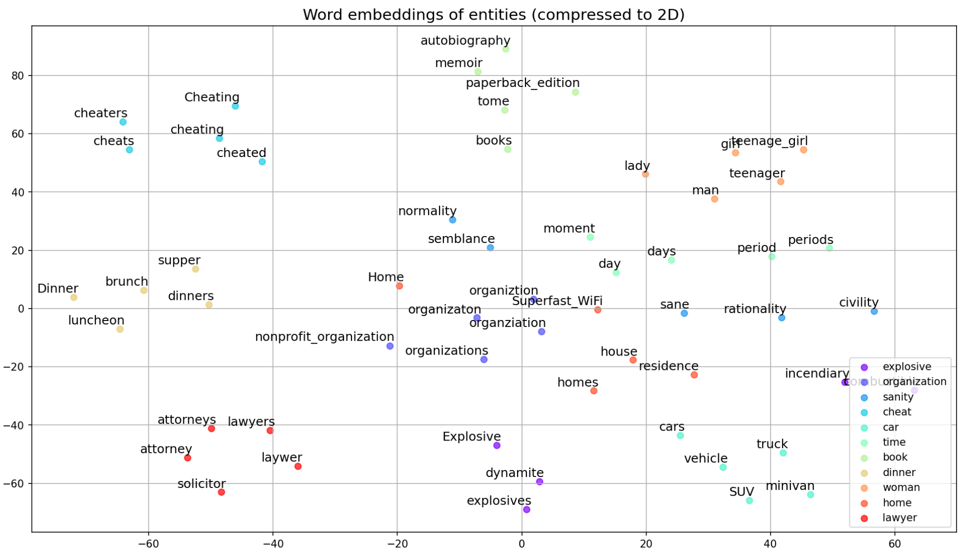


Figure 2: Generated Word Embeddings Clustered by Topic

5.2 Loss Function

The loss function for a given document is calculated as follows:

$$C = -(y_t * \sum_{i=1}^{n} log(p_i) + (1 - y_t) * \sum_{i=1}^{n} log(1 - p_i)) \quad (1)$$

Here, p_i is the coherence score (or probability) calculated for the i_{th} window of the current training post and y_t is the true coherence score(1 or 0).

The loss function aims to train the network such that it tends to drive the coherence score for a coherent post (indicated by $y_t = 1$) towards 1. On the flip side, the target coherence score for an incoherent post (indicated by $y_t = 0$) towards 0. The first part of the loss function becomes active for coherent posts and penalises smaller scores while the second part of the loss function becomes active for incoherent posts and penalises larger scores.

6 Experimental Results and Analysis

We initially trained our model on the 2 generated datasets explained in section 3. Since these datasets are new, we needed a method to compare our model with the existing architectures. Hence, we also trained our model on the Accidents Reports Dataset (Barzilay and Lapata, 2005). As mentioned ealier, it is one of the standard corpora used for text coherence. For the accident corpora, we use the same train-test split used by the previous architecture, i.e. 100 documents for training and 100 for testing with 20 permutations each for testing. The train-test split for the social-network generated dataset has been covered in section 3.3.

6.1 Comparison: Existing and New datasets

For each trained model, we tested on all the 3 testing sets to draw insights and analyse results from the models. The results are shown in Table 1.

We observe that the model trained on the Accident Reports dataset performs better on the Accident Reports test corpus giving an accuracy of 83.2% but attains a lower score on social media test datasets. On the other hand, the models trained on the 2 social media corpora perform better on the test sets from social media datasets than the accident reports test corpus. This can be attributed to the fact that social media and accident report corpora are different in terms of text and entities. Consequently, a model trained on one of the datasets tends to perform better on a similar test set. As future work, we could try training on one of the datasets and fine-tuning on the other to broaden the model's scope.

As mentioned in 3.2, in one social media dataset, the posts and comments are considered as separate documents, while in the other one posts and top 10 comments are considered as combined documents. The corpus with combined posts and comments performs better on all test sets in comparison to the separate social media dataset. This may be down to its ability to capture greater context from the corpus.

6.2 Comparison with existing models

To compare our design against existing state-of-the-art models, we trained our model on the accident reports dataset with the same train-test split as the

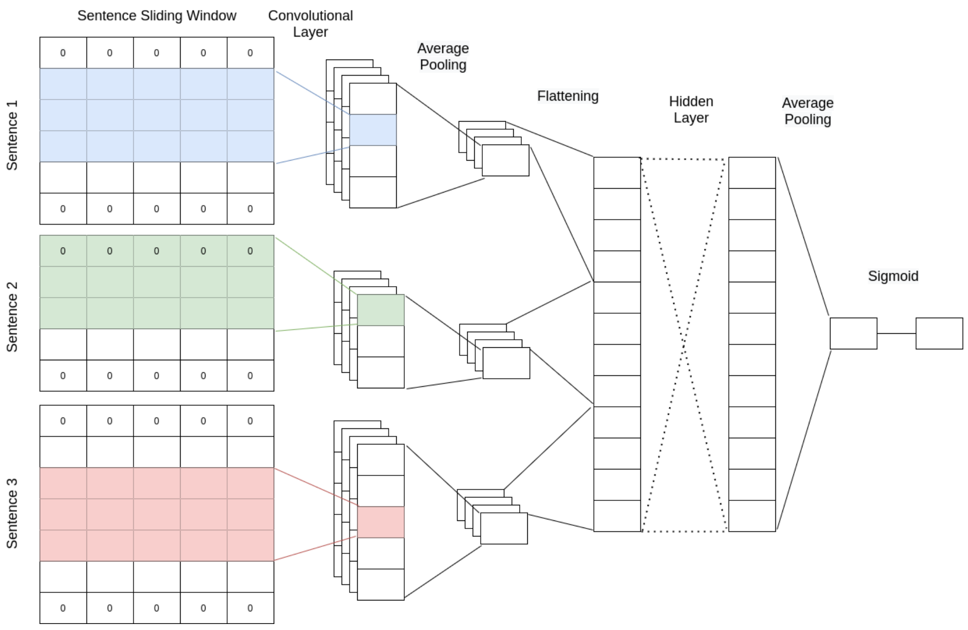


Figure 3: Proposed Sliding Window CNN Architecture

state-of-the-art models. The results are presented in Table 2.

The result values for prior works are taken from (Cui et al., 2017). Our model outperforms 2 of the existing text coherence models, namely, HMM+Content and Conference+Syntanx and is within 2% range of the Graph and Recurrent models. However, it lags behind the remaining state-of-the-art models. Minor improvements could be made by adjusting the learning rate or training for more epochs. Further, we could also consider using a similarity matrix layer like (Cui et al., 2017) to improve performance.

7 Conclusion and Future Work

In this project, we have developed a text coherence evaluation process for social media from end to end. First, we created a new dataset by scraping the top 1000 Reddit posts. Then, we pre-processed, cleaned and generated word embeddings for the corpus. Finally, we utilised a sliding window-based CNN to derive a text coherence score. We analysed the results of our model on existing datasets and also compared the model's performance on standard and generated datasets. Our results show that our architecture performs decently against existing literature, especially given that our model is exceptionally light-weight.

We could expand upon this work by trying to tailor our pre-processing more towards social media. For example, generating our own embeddings from social media texts could provide us with better results. Alternatively, we could also modify the architecture by skipping some connections to let the text be available much deeper into the network. Another approach we can explore is to train initially on the standard datasets and fine-tune the model later on social media datasets to get the best of both worlds.

8 Resource Links and Contribution

The following are the links to source code and generated datasets where outputs described in the paper can be seen:

- Dataset: Posts, comments separate.
- Dataset: Posts, top comments concatenated.
- Accident Report Dataset for text coherence.
- Notebook for dataset generation.
- Notebook for data preprocessing.
- Notebook for text coherence model.
- README for running the codes.

8.1 Individual Contribution

Dataset generation: Aman and Tushar wrote the code for extraction of posts and comments from Reddit Dataset. Udbhav and Mayank implemented the program for generating permutations for each document and storing the dataset.

Data Preprocessing: Aman and Tushar implemented the initial dataset loading, tokenization and later generating word embeddings from entities.

Table 1: Comparison among Existing and New Datasets

Training Set \ Testing Set	Accident	Social_Separate	Social_Combined
Accident	83.2	64.036	67.821
Social_Separate	64.85	71.107	70.786
Social_Combined	69.95	74.535	77.071

Table 2: Comparison with existing coherency models on NTSB accident Dataset

Name	Accuracy Percent
DCM (Cui et al., 2017)	95
Recursive (Li and Hovy, 2014)	86.4
Recurrent (Li and Hovy, 2014)	84
Graph (Guinaudeau and Strube, 2013)	84.6
HMM (Louis and Nenkova, 2012)	93.8
HMM+Content (Louis and Nenkova, 2012)	74.2
Entity Grid (Barzilay and Lapata, 2005)	90.4
Conference+Syntax (Barzilay and Lapata, 2005)	76.5
Our Model	83.2

Udbhav and Mayank wrote the code for remaining preprocessing like stemming, lemmatization, etc. and used this preprocessed data to extract topic-based entities using LDA.

Coherence Model: Udbhav and Mayank wrote the code for coherence model architecture and tested the designed model. Aman and Tushar wrote the code for training the designed model and developed the custom loss function.

Acknowledgments

We would like to acknowledge Dr. Ashish Anand and Dr. Amit Awekar for their valuable feedback, teaching and guidance throughout the course.

References

Regina Barzilay and Mirella Lapata. 2005. Modeling local coherence: An entity-based approach. In *43rd Annual Meeting on Association for Computational Linguistics*, page 141–148.

Baiyun Cui, Yingming Li, Yaqing Zhang, and Zhongfei Zhang. 2017. Text coherence analysis based on deep neural network.

Micha Elsner and Eugene Charniak. 2008. Coreference-inspired coherence modeling. In *Proceedings of ACL-08: HLT, Short Papers*, pages 41–44.

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

Yuan Feng Jelodar, Wang. Latent dirichlet allocation (lda) and topic modeling: models, applications, a survey.

Jiwei Li and Eduard Hovy. 2014. A model of coherence based on distributed sentence representation. In *2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 2039–2048.

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

Annie Louis and Ani Nenkova. 2012. A coherence model based on syntactic patterns. In *2012 Joint Conference on Empirical Methods in NLP and CNLL*, pages 1157–1168.

Mohsen Mesgar and Michael Strube. 2016. Lexical coherence graph modeling using word embeddings. In *2016 Conference of the North American Chapter of the ACL: Human Language Technologies*, pages 1414–1423.

Mohsen Mesgar and Michael Strube. 2018. A neural local coherence model for text quality assessment. In *2018 Conference on Empirical Methods in Natural Language Processing*, pages 4328–4339.