Abstract

Coherence is established by semantic connections between sentences of a text which can be modeled by lexical relations. In this paper, we introduce the lexical coherence graph (LCG), a new graph-based model to represent lexical relations among sentences. The frequency of subgraphs (coherence patterns) of this graph captures the connectivity style of sentence nodes in this graph. The coherence of a text is encoded by a vector of these frequencies. We evaluate the LCG model on the readability ranking task. The results of the experiments show that the LCG model obtains higher accuracy than state-of-the-art coherence models. Using larger subgraphs yields higher accuracy, because they capture more structural information. However, larger subgraphs can be sparse. We adapt Kneser-Ney smoothing to smooth subgraphs’ frequencies. Smoothing improves performance.

1 Introduction

The concept of coherence is based on cohesive semantic relations connecting elements of a text. Cohesive relations are expressed through grammar and the vocabulary of a language. The former is referred to as grammatical coherence, the latter as lexical coherence (Halliday and Hasan, 1976). Grammatical coherence encompasses coreference, substitution, ellipsis, etc. Lexical coherence comprises semantic connections among words of a text.

In this paper we measure text coherence by modeling lexical coherence. Lexical relations specify cohesive relations over the sentences of a text. These lexical relations can be any kind of semantic relation: repetition, synonymy, hyperonymy, meronymy, etc. These lexical items may or may not have the same reference (Halliday and Hasan, 1976).

Why does the little boy wriggle all the time? Girls don’t.

In this example the lexical items boy and girls are semantically related. Although they do not refer to the same entity, they still connect these two sentences.

There is coherence between any pair of lexical items that stand to each other in some lexicosemantic relation (Halliday and Hasan, 1976). For textual purposes it is not required to determine the type of the relation. It is only necessary to recognize semantically related lexical items, and these relations can be learned by cooccurring lexical items.

One can use world knowledge resources to determine semantic relations. This way is expensive in terms of determining the best resource, e.g. WordNet vs. Freebase. WordNet lacks broad coverage in particular with proper names, Freebase is restricted to nominal concepts and entities.

Recent improvements in embedding representations of words let us efficiently compute semantic relations among lexical items in the vector space. These models use a vector of numbers to encode the meaning of words. We use these vectors to check the existence of any kind of semantic relations between two words.

In the following example the sentences are connected because of the semantic relation between king and queen which can be induced by word em-
bedding models (Mikolov et al., 2013; Pennington et al., 2014).

...The king was in his counting-house, counting out his money,
The queen was in the parlour, eating bread and honey.

We model lexical coherence between sentences by a lexical coherence graph (LCG). We consider subgraphs of this graph coherence patterns and use their frequency as features representing the connectivity of the graph and, hence, the coherence of a text (Mesgar and Strube, 2015).

An important task for evaluating a coherence model is readability assessment. The goal of this task is to rate texts based on their readability. The more coherent a text, the faster to read and easier to understand it is. Other coherence models (Barzilay and Lapata, 2008; Guinaudeau and Strube, 2013; Mesgar and Strube, 2014) are also evaluated on this task. Pitler and Nenkova (2008) use the entity grid (Barzilay and Lapata, 2008) to capture the coherence of a text for readability assessment. Mesgar and Strube (2015) extend the entity graph (Guinaudeau and Strube, 2013) as coherence model to measure the readability of texts. They encode coherence as a vector of frequencies of subgraphs of the graph representation of a text. We build upon their method and represent the connectivity of sentences in our LCG model by a vector of frequencies of subgraphs.

Although using the frequency of subgraphs of the lexical coherence graph encodes coherence features well, the subgraph frequency method, in general, is suffering from a sparsity problem when the subgraphs get larger. Large subgraphs capture more structural information, but they occur only rarely. We resolve this sparsity issue by adapting Kneser-Ney smoothing (Heafield et al., 2013) to smooth subgraph counts (Section 3). We estimate the probability of unseen subgraphs, i.e. coherence patterns. This prediction lets us measure the coherence of a text even when its corresponding graph representation contains a subgraph which does not occur in the training data. If the unseen coherence pattern is similar to seen ones, smoothing gives it closer probability to seen coherence patterns in comparison to dissimilar unseen ones. This is due to the base probability factor in Kneser-Ney smoothing.

We evaluate our LCG model on the two readability datasets provided by Pitler and Nenkova (2008) and De Clercq et al. (2014), respectively (Section 4). The results (Section 5) indicate that the LCG model outperforms state-of-the-art systems. By applying Kneser-Ney smoothing we solve the sparsity problem. Smoothing allows us to exploit the high informativity of large subgraphs which leads to new state-of-the-art results in readability assessment.

2 Related Work

The entity grid model (Barzilay and Lapata, 2008) is based on entity transitions over sentences. It uses a two dimensional matrix to represent transitions of entities among adjacent sentences. The entity grid is applied to readability assessment by Pitler and Nenkova (2008). The entity graph (Guinaudeau and Strube, 2013) is a graph-based, mainly unsupervised interpretation of the entity grid. This model represents the distribution of entities over sentences in a text with a bipartite graph. Connections between sentences are obtained by information on entities shared by sentences. Guinaudeau and Strube (2013) perform a one-mode projection on sentence nodes and use the average out-degree of the one-mode projection graph to quantify the coherence of the given text. Mesgar and Strube (2015) represent the connectivity of the one-mode projection graph by a vector whose elements are the frequencies of subgraphs in projection graphs. This encoding works much better than the entity graph for the readability task on the P&N dataset and even outperforms Pitler and Nenkova (2008) by a large margin. Zhang et al. (2015) state that the entity graph model is limited, because it only captures mentions which refer to the same entity (the entity graph uses a very restricted version of coreference resolution to determine entities). Zhang et al. (2015) use world knowledge YAGO (Hoffart et al., 2013), WikiPedia (Denoyer and Gallinari, 2006) and FreeBase (Bollacker et al., 2008) to capture the semantic relatedness between entities even if they do not refer to the same entity. Main issues with using world knowledge are: the choice knowledge sources, selection of knowledge from the source, coverage, and language-dependence.

Word embedding approaches like word2vec and
GloVe (Mikolov et al., 2013; Pennington et al., 2014) show that the semantic connection between words can be captured by word vectors which are obtained by applying a neural network. The ability to train on very large data sets allows the model to learn complex relationships between words.

3 Method

We introduce a new graph representation of semantic connections over lexical items in texts. Afterwards we compute the frequency of all subgraphs, i.e. coherence patterns. The intuition is that subgraphs capture how sentence nodes are connected and, respectively, encode text coherence.

3.1 Graph Model

We model semantic relations between sentences by a graph \( G = \langle V, E \rangle \) where \( V \) is the set of sentence nodes and \( E \) is the set of edges between sentence nodes. Two nodes of \( G \) are adjacent if there is a semantic connection between the corresponding sentences. Two sentences are semantically connected if there is at least one strong semantic relation between the words of these sentences. We model semantic relations between words by their corresponding word embeddings (Pennington et al., 2014). Given word vectors \( v_a \) for word \( a \) of sentence \( A \) and \( v_b \) for word \( b \) of sentence \( B \), the cosine similarity value, \( \cos(v_a, v_b) \), between the two word vectors is a measure of semantic connectivity of the two words. The range of \( \cos(v_a, v_b) \) is between \([-1, +1]\). One interpretation of cosine is the normalized correlation coefficient, which states how well the two words are semantically correlated (Manning and Schütze, 1999). The absolute value of cosine, \(|\cos(v_a, v_b)|\), encodes how strongly the two words are connected.

The connection between sentences is obtained from connections between their words (Figure 1). Assume sentence \( A \) precedes sentence \( B \), each word \( b \) of sentence \( B \) is connected with word \( a^* \) of \( A \), where

\[
a^* = \arg\max_a \cos(b, a)
\]

Then from all connections between the words of sentences \( A \) and \( B \), the connection with the maximum weight among the words of \( B \) is selected to connect these two sentences (Figure 2).

The output of this phase is a graph whose edge weights model the strength of connections between sentences. The edges in this graph are directed to model the order of sentences.

Word embeddings relate each word in sentence \( A \) with each word in sentence \( B \). Since the resulting graph is very dense, we filter out edges whose weights are below a threshold\(^1\).

3.2 Coherence Features

Mesgar and Strube (2015) propose that the connection style of an entity graph can be captured by the frequency of all \( k \)-node subgraphs in this graph. Larger\(^2\) subgraphs\(^3\) can capture more information about the structure of graphs and are more informative coherence patterns than smaller ones. We experiment with \( k \in \{3, 4, 5, 6\} \). Text coherence is repre-

\(^1\)We set this threshold to 0.9 to connect only sentences with high confidence.

\(^2\)The size of a subgraph is the number of its nodes.

\(^3\)We compute induced subgraphs (Mesgar and Strube, 2015). However, we use the term subgraph for brevity.
sented by a vector whose elements are the frequency of subgraphs (coherence patterns) with $k$-node.

### 3.3 Smoothing

Although increasing the size $k$ of subgraphs captures more structural information about the connections of sentence nodes, a main risk with large subgraphs is sparsity. Given a sentence graph, many large subgraph types do not occur in this graph. Small subgraph types occur frequently in most sentence graphs in the dataset, but these subgraphs do not capture enough information about the connectivity style of the graphs.

Inspired by Kneser-Ney smoothing in language models (Heafield et al., 2013), each feature vector of a sentence graph can be smoothed. Smoothing deals with the problem of zero counts in the feature vector. It also lets the model having feature values for unseen subgraphs (like OOV in language modeling) which may be seen in the testing phase.

Kneser-Ney smoothing uses a discount factor to discount the raw count of each event (subgraph) and distributes the total discount to all event (subgraph) probabilities by means of a base probability.

The estimated frequency of subgraph $sg$ in a given sentence graph is computed as follows:

$$\text{KN}(sg) = \frac{\max\{\text{count}(sg) - \alpha, 0\}}{Z} + \frac{M \cdot \alpha}{Z} P_b(\text{sg}),$$

where $\alpha$ is the discount factor and $M$ is the number of times that discount factor is applied. $Z$ is a normalization factor to ensure that the distribution sums to one and is obtained as follows:

$$Z = \sum_{sg \in A} \text{count}(sg),$$

where $A$ is the set of all subgraphs with $k$-nodes and function count$(\cdot)$ computes the number of instances of subgraph $sg$ in the given sentence graph.

$P_b(\text{sg})$ in Kneser-Ney smoothing is the base probability of subgraph $sg$ among all $k$-node subgraphs (A). The base probability can be computed based on hierarchical (parent-child) relations in subgraphs. $k$-node subgraph $sg_i$ is a parent of $(k+1)$-node subgraph $sg_j$, if $sg_i$ is a subgraph of $sg_j$. Figure 3 shows the parent-child relation between subgraphs via a weighted tree. The root of this tree is a null graph$^4$. The weight of a parent-child relation connecting the parent subgraph $sg_i$ and child subgraph $sg_j$ is shown by $w_{ij}$ and computed as follows:

$$w_{ij} = \frac{\text{count}(sg_i, sg_j)}{\sum_{sg \in A} \text{count}(sg_i, sg_j)},$$

where $A$ is all subgraphs with $k$-node and $k$ equals the number of nodes of $sg_j$. Interpretation of weight $w_{ij}$ is the normalized count of $sg_i$ in $sg_j$ with respect to all outgoing edges from $sg_i$.

The base probability of each subgraph $sg_j$ is the inner product of the Kneser-Ney probabilities of $sg_j$’s parents by the weights of the corresponding relations:

$$P_b(sg_j) = P \cdot W,$$

where $P$ is the vector of probabilities of all parents of $sg_j$ and $W$ is the vector of all corresponding edge weights connecting the parents of $sg_j$ to $sg_j$.

Since the root node of this tree is the null subgraph, and it is a subgraph of all possible sentence graphs, its base probability is one. Because the edge weights are in the range $[0, 1]$ the sum of the probabilities of all subgraphs with $k$-node is always equal to one.

**Proof.** Assume $I$ and $J$ are the set of all $k$-node and $(k+1)$-node subgraphs. We also assume that $I$ has $n$ subgraphs and $\sum_{i=1}^{n} p(sg_i) = 1$. Considering these assumptions we prove that

$$\sum_{j=1}^{m} p(sg_j) = 1,$$

where $m$ is the number of subgraphs in $J$.

We start from the left and compute the value of

$$\sum_{j=1}^{m} p(sg_j).$$

Based on the definition of base probability, the value of $p(sg_j)$ is computed based on its parents in $I$,

$$p(sg_j) = \sum_{i=1}^{n} w_{ij}p(sg_i),$$

where $w_{ij}$ is the weight of the parent-child relation between $sg_i$ and $sg_j$. Now we have:

$^4$A null graph is a graph with no nodes.
If we exchange the place of the sums and re-write the equation, we have:

\[ \sum_{j=1}^{m} p(sg_j) = \sum_{i=1}^{n} \sum_{j=1}^{m} w_{ij} p(sg_i). \]

In this equation \( p(sg_i) \) is independent of \( j \) (index of the inner sum), so it can be moved out of the inner sum:

\[ \sum_{j=1}^{m} p(sg_j) = \sum_{i=1}^{n} p(sg_i) \sum_{j=1}^{m} w_{ij}. \]

The inner sum equals 1.

\[ \sum_{j=1}^{m} p(sg_j) = \sum_{i=1}^{n} p(sg_i). \]

Based on our assumption the right side of the equation is 1 and

\[ \sum_{j=1}^{m} p(sg_j) = 1. \]

So we proved that the sum of the base probability of all \( k\)-node subgraphs is 1. \( \square \)

This way, Kneser-Ney smoothing distributes the total discount value by considering the weights of parent-child relations among the subgraphs. The result of applying smoothing is an estimation of the frequency of each subgraph in the sentence graph.

4 Experiments

4.1 Evaluation Task

We evaluate our coherence model on the task of ranking texts by their readability. The intuition is that more coherent texts are easier to read.

Datasets. We run our experiments on two datasets annotated with readability information provided by human annotators: \( P\&N \) (Pitler and Nenkova, 2008) and \( De\ Clercq \) (De Clercq et al., 2014).

The dataset \( P\&N \) contains 27 articles randomly selected from the Wall Street Journal corpus\(^5\). The average number of sentences is about 10 words. Every article is associated with a human score between \([0.0, 5.0]\) indicating the readability score of that article. We create pairs of documents, if the difference between their readability scores is greater than 0.5. If the first document in a pair has the higher score, we label this pair with +1, otherwise with −1. The resulting number of text pairs in this dataset is 209.

The dataset \( De\ Clercq \) consists of 105 articles from different genres: administrative (17 articles), journalistic (43 articles), manuals (14 articles) and miscellaneous (31 articles). The average number of sentences is about 12. This dataset was annotated by De Clercq et al. (2014) by asking human judges to compare two texts based on their readability. They use five labels:

\(^5\)Pitler and Nenkova (2008)'s dataset contains 30 articles. They remove one. We assume this is \texttt{waj-0382} which does not exist in the Penn Treebank. We furthermore remove \texttt{waj-2090} which does not exist in the final release of the Penn Discourse Treebank. We also remove \texttt{waj-1398} which is a poem and, hence, not very informative for readability assessment.
LME: left text is much easier,
LSE: left text is somewhat easier,
ED: both texts are equally difficult,
RSE: right text is somewhat easier,
RME: right text is much easier.

We map these labels to three class labels:
+1: for text pairs where the left text is easier to read (LME or LSE),
0: for text pairs where both texts are equally difficult to read (ED),
−1: for text pairs where the right text is easier to read (RSE or RME).

Properties of this dataset are shown in Table 1.

<table>
<thead>
<tr>
<th>Genre</th>
<th>No. of articles</th>
<th>No. of text pairs</th>
</tr>
</thead>
<tbody>
<tr>
<td>Administrative</td>
<td>17</td>
<td>272</td>
</tr>
<tr>
<td>Journalistic</td>
<td>43</td>
<td>1806</td>
</tr>
<tr>
<td>Manuals</td>
<td>14</td>
<td>182</td>
</tr>
<tr>
<td>Miscellaneous</td>
<td>31</td>
<td>931</td>
</tr>
</tbody>
</table>

Table 1: Properties of the different genres in the De Clercq dataset.

4.2 Experimental Settings

Word Embeddings and Classification. In order to reduce the effect of very frequent words, stop words are filtered by using the SMART English stop word list (Salton, 1971). We use a pretrained model of GloVe for word embeddings. This model is trained on Common Crawl with 840B tokens, 2.2M vocabulary. We represent each word by a vector with length 300 (Pennington et al., 2014). For handling out-of-vocabulary words, we assign a random vector to each word and memorize it for its next occurrence (Kusner et al., 2015). The classification task is done by the SVM implementation in WEKA (SMO) with the linear kernel function. All settings are set to the default values. The evaluation is computed by 10-fold cross validation.

Graph Processing and Smoothing. In order to compare the performance of LCG with the entity graph model, we follow Mesgar and Strube (2015) and use the gSpan method (Yan and Han, 2002) to compute all common subgraphs on each dataset and their frequencies. Note that gSpan does not count all possible k-node subgraphs, whereas for applying Kneser-Ney smoothing it is necessary to count all possible k-node subgraphs, because the probability should be distributed among all possible subgraphs. This also helps to estimate the probability of unseen patterns. We use a random sampling method (Shervashidze et al., 2009) to obtain the frequency of subgraphs in a sentence graph. In this regard, we take 10,000 samples of the given sentence graph by randomly selecting k nodes of the graph to count the occurrence of k-node subgraphs in this graph. We compute the base probability for at most k = 6. We find the best value for d in a greedy manner. First, we initialize d with 0.001. In each iteration we compute the performance. Then we multiply the discount factor by 10. We iterate as long as the discount factor is less than 1000. We report the best performance.

5 Results

In order to compare our method with related work, we run our model on the P&N dataset. Table 2 reports the accuracy of LCG with different values for k in k-node subgraphs. This corresponds to coherence patterns spanning different numbers of sentences.

<table>
<thead>
<tr>
<th>System</th>
<th>Accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>ZeroR</td>
<td>50.24%</td>
</tr>
<tr>
<td>EGrid</td>
<td>83.25%</td>
</tr>
<tr>
<td>k-node EGraph</td>
<td></td>
</tr>
<tr>
<td>k-node EGraph+PRN</td>
<td></td>
</tr>
<tr>
<td>k-node LCG</td>
<td></td>
</tr>
<tr>
<td>3-node</td>
<td>79.43%</td>
</tr>
<tr>
<td>4-node</td>
<td>89.00%</td>
</tr>
<tr>
<td>5-node</td>
<td>96.17%**</td>
</tr>
<tr>
<td>3-node EGraph</td>
<td>80.38%**</td>
</tr>
<tr>
<td>4-node EGraph+PRN</td>
<td>89.95%</td>
</tr>
<tr>
<td>5-node LCG</td>
<td>95.69%**</td>
</tr>
<tr>
<td>3-node EGraph+PRN</td>
<td>78.95%</td>
</tr>
<tr>
<td>4-node EGraph+PRN</td>
<td>89.47%</td>
</tr>
<tr>
<td>5-node LCG</td>
<td>97.13%</td>
</tr>
</tbody>
</table>

Table 2: P&N dataset.
and slightly worse for 5-node subgraphs than the EGraph. The lexical coherence graph model, LCG, performs slightly worse than EGraph on 3-node subgraphs. This could be because the graphs in LCG have more edges than the graphs in EGraph. When graphs are denser 3-node subgraphs occur in every graph, hence their frequency is less discriminative. As shown in Table 2 larger subgraphs (4-node and 5-node) capture more information and improve upon EGraph and for 5-node subgraphs even upon EGraph+PRN. LCG significantly (p-value = 0.01) works better than EGraph+PRN and EGraph using 5-node subgraphs. The difference between LCG and EGraph+PRN and EGraph using 4-node subgraphs is not significant.

Table 3 shows the performance of different models on the De Clercq dataset.

<table>
<thead>
<tr>
<th>System</th>
<th>Accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>ZeroR</td>
<td>42.312%</td>
</tr>
<tr>
<td>k-node</td>
<td>EGraph+PRN</td>
</tr>
<tr>
<td>3-node</td>
<td>42.31%</td>
</tr>
<tr>
<td>4-node</td>
<td>48.07%</td>
</tr>
<tr>
<td>5-node</td>
<td>65.77%</td>
</tr>
</tbody>
</table>

Table 3: De Clercq dataset.

Again, we use a majority baseline (ZeroR) to put our results in context. While the performance of both methods almost does not beat the baseline for 3-node subgraphs, 4-node subgraphs work already better, and 5-node subgraphs yield reasonable performance on this dataset. Although EGraph+PRN and LCG reach almost the same performance for 4-node, the difference between them is statistically significant (p-value = 0.01). With 5-node subgraphs, LCG outperforms EGraph+PRN subgraphs by a large margin and gets a very reasonable performance on this dataset.

The general performance on the De Clercq dataset is lower than the performance on the the P&N dataset. This can have two reasons: first, the ranking task on the De Clercq dataset is three-label classification which is more difficult than the binary classification task on the P&N dataset. Second, texts in the De Clercq dataset are from different genres and coherence patterns may vary across genres. Hence, we take a closer look on the performance on the different genres.

<table>
<thead>
<tr>
<th>5-node</th>
<th>EGraph+PRN</th>
<th>LCG</th>
</tr>
</thead>
<tbody>
<tr>
<td>Admin</td>
<td>69.49%</td>
<td>71.69%</td>
</tr>
<tr>
<td>Jur</td>
<td>65.01%</td>
<td>82.12%</td>
</tr>
<tr>
<td>Manual</td>
<td>54.95%</td>
<td>61.54%</td>
</tr>
<tr>
<td>Misc.</td>
<td>70.68%</td>
<td>76.69%</td>
</tr>
</tbody>
</table>

Table 4: Accuracy of EGraph+PRN and LCG on different genres in the De Clercq dataset.

Table 4 shows the performance for EGraph+PRN and LCG using 5-node subgraphs on the different genres in the De Clercq dataset. The performance of LCG is higher than EGraph+PRN on all genres. Unlike EGraph+PRN, LCG gets the best performance on journalistic articles. The lowest performance of both models is obtained on manuals. On administrative articles, performance of LCG is slightly better than EGraph+PRN. On miscellaneous articles LCG performs better than EGraph+PRN.

While large subgraphs are very informative for coherence modeling, extracting large subgraphs (k > 4) in relatively small datasets leads to a data sparsity problem, as there are very many possible subgraphs to be represented in a high dimensional vector space. Hence, many possible subgraphs have low or even zero counts. The problem for such a vector is that each graph is only similar to itself and not to any other graph. Hence, we observe a drop in performance when the model deals with large subgraphs (6-node subgraphs, LCG1 for P&N in Table 5). We solve this problem by smoothing.

In order to apply Kneser-Ney smoothing we use a sampling method to create all possible (connected and disconnected) k-node subgraphs (for LCG1 and LCG1* we use connected and disconnected subgraphs, for LCG only connected ones).

Table 5 shows the performance of LCG1 when it is applied to ever larger subgraphs. As can be seen in Table 5, the performance on the P&N dataset suddenly drops for 6-node subgraphs. This is could be caused by the sparsity problem.

When we apply Kneser-Ney smoothing as described in Section 3 the results for all tested values of k are superior for LCG1* when compared to LCG1 (Table 5).

Kneser-Ney smoothing improves the performance of the system even with 3-node subgraphs by a large margin. Smoothing reduces the power of fre-
frequency and makes the frequency distribution of subgraphs more even. Smoothing reduces the values through all subgraphs by considering parent-child relations between subgraphs to relate similar subgraphs. That is the advantage of the Kneser-Ney method in comparison to the other smoothing methods like Laplace-Smoothing.

For the P&N dataset we achieve the best results to date. Pitler and Nenkova (2008) reported 83.25% accuracy, Mesgar and Strube (2015) 89.95%. When smoothing 5-node subgraphs we are able to report 98.08%. This, however, indicates that this dataset may not be the best one to report performance on. Hence, we now check whether smoothing also improves the performance on the more difficult De Clercq dataset.

On this dataset, we basically observe the same trends. Both settings result in better performance than LCG (see Table 3).

Note that none of the parameters in this work is tuned on the datasets. One may get better performance by tuning the parameters. The results confirm the intuition that the lexical coherence graph LCG captures coherence and models lexical coherence appropriately.

Applying smoothing on graphs of EGraph+PRN model increases the performance of this model. But this improvement is not as high as the improvement on the LCG graph.

Coherence Patterns. In this part we check the Pearson correlation coefficient between LCG1 and human judgements of a few frequent subgraphs on the P&N dataset. In order to be consistent with Mesgar and Strube (2015), we use the exhaustive value of subgraph frequencies, i.e. LCG1 for our work.

For the 3-node subgraphs only one subgraph (Figure 4) in the LCG1 representation is significantly (and positively) correlated (p-value < 0.05) with human scores. For the 4-node subgraphs, we find six subgraphs which are significantly correlated with readability. Only one is positively correlated, while four are negatively correlated. Interestingly, both positively correlated 3-node and 4-node subgraphs have been determined as positively and significantly correlated by Mesgar and Strube (2015) as well. Both also capture a similar coherence pattern, indicating that our method is linguistically sound.

<table>
<thead>
<tr>
<th>Pattern</th>
<th>ρ</th>
<th>p-value</th>
</tr>
</thead>
<tbody>
<tr>
<td>3-node</td>
<td>0.43</td>
<td>0.024</td>
</tr>
<tr>
<td>4-node</td>
<td>-0.45</td>
<td>0.018</td>
</tr>
</tbody>
</table>

Figure 4: Pearson correlation between 3-node and 4-node subgraphs and readability scores in the P&N dataset.

6 Conclusions and Future Work

In this paper we propose a new graph based coherence model, the lexical coherence graph, LCG. We view coherence as semantic connectedness between words which we model by word embeddings. We take only the strongest connection between sentences to create a graph with connected sentences. Then we extract large subgraphs capturing coherence patterns, which show similarity to patterns described in text linguistics (Daneš, 1974).
While the entity grid works only on sequences of up to three adjacent sentences, we are able to model relationships of up to six non-adjacent sentences. We solve the sparsity problem of large subgraphs by adapting Kneser-Ney smoothing to graphs. Smoothing prevents LCG from losing performance with large subgraphs and leads to superior performance on the Pitler and Nenkova (2008) dataset and to a first reasonable state-of-the-art on the De Clercq et al. (2014) dataset.

In future work we want to apply LCG to essay scoring as well. Also, we see that our adaption of Kneser-Ney smoothing to graphs may be useful for research in subgraph mining in general.

Acknowledgments

This work has been funded by the Klaus Tschira Foundation, Heidelberg, Germany. The first author has been supported by a HITS Ph.D. scholarship. We would like to thank Orphée De Clercq who provided the De Clercq dataset. We also appreciate Andreas Spitz’ comments on graph processing.

References


Xifeng Yan and Jiawei Han. 2002. gSpan: Graph-based substructure pattern mining. In *Proceedings of the International Conference on Data Mining*, Maebashi City, Japan, 9–12 December 2002, pages 721–724.