Learning to Identify Sentence Parallelism in Student Essays

Wei Song[†], Tong Liu[†], Ruiji Fu[‡], Lizhen Liu[†], Hanshi Wang[†], Ting Liu[§]
[†]Information Engineering, Capital Normal University, Beijing
[‡]Iflytek Research Beijing, Beijing
[§]Harbin Institute of Technology, Harbin
{wsong, tongliu, lzliu, hswang}@cnu.edu.cn, rjfu@iflytek.com, tliu@ir.hit.edu.cn

Abstract

Parallelism is an important rhetorical device. We propose a machine learning approach for automated sentence parallelism identification in student essays. We build an essay dataset with sentence level parallelism annotated. We derive features by combining generalized word alignment strategies and the alignment measures between word sequences. The experimental results show that sentence parallelism can be effectively identified with a F_1 score of 82% at pair-wise level and 72% at parallelism chunk level. Based on this approach, we automatically identify sentence parallelism in more than 2000 student essays and study the correlation between the use of sentence parallelism and the types and quality of essays.

1 Introduction

Parallelism is an important rhetorical device. It can be defined as *two or more coherent text spans (phrases or sentences), which have similar syntactic structures and related semantics, and express relevant content or emotion together.* Each text span is a parallelism unit and the parallel units form a parallelism chunk. The following two sentences segmented by the semicolon form an example of sentence parallelism.

The inherent vice of capitalism is the unequal sharing of blessing;

the inherent virtue of socialism is the equal sharing of miseries. by Churchill.

Parallelism adds balance and rhythm to make speeches and writings more vivid and powerful. Moreover, parallelism also adds clarity to the sentence or even the discourse. Several sentences are expressed similarly to show that the content in the sentences are equal in importance. Therefore, properly using parallelism may improve the quality of texts. On the other hand, identifying parallelism in essays would potentially help to evaluate the quality of writings and benefit applications like essay scoring and organization evaluation.

In this paper, we study the problem of identifying parallelism in student essays. We focus on identifying sentence parallelism. A parallelism unit is a sentence, and several *parallel sentences* form a *parallelism chunk*. Parallelism identification is a task to find the parallelism chunks within essays.

This task is nontrivial. There are several factors to be considered. Since the parallel sentences should have similar structures and related semantics, they can be seen as forming a certain kind of alignment between each other. However, such alignment can exist in various levels, from surface lexical patterns to syntactic structures, semantics and even emotions. Moreover, the alignment can occur at various granularity (words, phrases, clauses or sentences). Therefore, it is difficult to design manual rules to identify sentence parallelism.

We propose a learning based framework for sentence parallelism identification. We annotate a sentence parallelism dataset consisting of about 500 student essays. This dataset allows us to derive features to model sentence parallelism and utilize machine learning to learn a prediction model. Since parallelism can be seen as a kind of alignment, we study various alignment measures to quantify the alignment between sentences. Sentence alignment depends on word alignment so that we exploit several strategies to generalize word alignment based on semantic and syntactic properties. The interactions among alignment measures and word alignment strategies generate features to represent the alignment between sentences. The experimental results show that sentence parallelism can be effectively identified. The F_1 score can reach 82% at pair-wise level and 72% at parallelism chunk level. The features based on different alignment measures and different word alignment strategies complement each other. We further study the use of sentence parallelism in more than 2000 student essays based on automated sentence parallelism identification. We observe that the use of sentence parallelism varies in narrative and argumentative essays and has a positive correlation to the quality of writings especially in argumentative essays.

Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers, pages 794–803, Osaka, Japan, December 11-17 2016.

2 Data

We collected student essays written by Chinese students from a senior high school during mock examinations. The essay types include narrative and argumentative essays, covering multiple topics. Two labelers were asked to label parallelism in randomly sampled essays at sentence level. They were guided by the definition of parallelism. Sentences are obtained by the sentence splitter provided by the Chinese language processing toolkit — HIT-LTP (Che et al., 2010). If a sentence contains less than four words, it is not allowed to be labeled. A sentence parallelism chunk consists of multiple sentences. The labelers recognized the sentence parallelism chunks in essays and assigned a distinct number to all parallel sentences from the same chunk in order to distinguish different chunks.

Item	Number
#Essay	544
avg. #sentence per essay	28.47
avg.#parallelism chunk per essay	2.03
avg. #sentence per chunk	2.68

Table 1: Statistics of the annotated sentence parallelism dataset.

After annotation, we collected 544 student essays, each of which has at least one sentence parallelism chunk. 30 essays were annotated by both labelers, and the Kappa value between them is 0.71 (Carletta, 1996), which indicates a moderate consistence. The mainly disagreement lies in their different judgement standards in terms of the quality of parallelism between sentences. After discussion and reaching a consensus, they reviewed all the annotations. Table 1 shows the basic statistics of the dataset.

3 Sentence Parallelism Identification

We cast sentence parallelism identification as a classification problem. Given an essay, we conduct a binary classification for every pair of sentences to determine whether they are *parallel* or *non-parallel*. Further, we get parallelism chunks according to the results of pair-wise classification.

According to the definition of parallelism, parallel sentences are expected to have sorts of alignment. The alignment can be about words, syntactic structures and semantics. In this section, we would exploit a set of alignment measures to quantity sentence alignment for a pair of sentences. For all alignment measures, the basis is the alignment between single words. Therefore, we start by studying word alignment strategies and look forward to incorporate information from multiple aspects. Then we introduce the alignment measures that are adapted for this task. Combining alignment measures and word alignment strategies, we can derive rich features to represent sentence alignment.

We use the HIT-LTP toolkit (Che et al., 2010) to conduct word segmentation, part-of-speech (POS) tagging and dependency parsing. All alignment measures would be computed on word sequences.

3.1 Word Alignment Strategies

Without loss of generality, we define a matrix R, which measures the alignment between every pair of tokens in vocabulary $\mathcal{V}$. $R(w, v)$ represents the alignment score between a pair of tokens (w, v), $w, v \in \mathcal{V}$. According to different assumption, $R(w, v)$ may have different values. We consider the following word alignment strategies:

- **Exact Match**: $R(w, v) = 1$, if $str(w) == str(v)$, otherwise $R(w, v) = 0$. $str(w)$ is the surface string of w.
- **POS Match**: $R(w, v) = 1$, if $pos(w) == pos(v)$, otherwise $R(w, v) = 0$. $pos(w)$ is the POS tag of word w.
- **Syntactic Role Match**: $R(w, v) = 1$, if $syntacticrole(w) == syntacticrole(v)$, otherwise $R(w, v) = 0$. $syntacticrole(w)$ is the syntactic role of word w. Here, we use dependency parsing to get the dependency labels as the syntactic roles. Considering the example shown in Figure 1, $syntacticrole$(春风)=SBV, $syntacticrole$(摇曳)=HED and $syntacticrole$(大地)=VOB.
- **Semantic Match**. $R(w, v) = 1$, if $similarity(w, v) > threshold$, otherwise $R(w, v) = 0$. $similarity(w, v)$ is a semantic similarity measure for w and v. Here, we compute the similarity based on word embeddings. Word embeddings are distributed representations of words learned on large scale corpus using neural networks (Mikolov et al., 2013). Each word is represented by a dense real value vector. $similarity(w, v)$ computes the cosine similarity between the vectors of w and v. The threshold is empirically set to 0.75.

The above strategies generalize words to various levels. We expect such generalization could help find alignment between sentences. For example, consider the following parallel sentences:

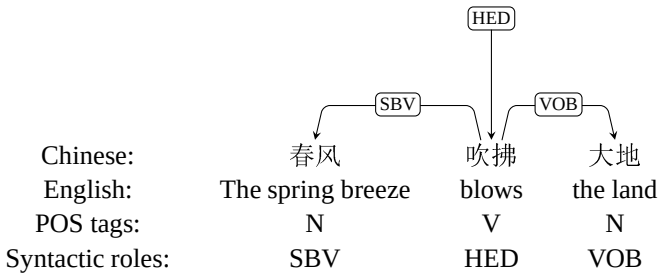


Figure 1: Dependency parsing tree for the sentence *春风吹拂大地(The spring breeze blows the land).*

春风吹拂大地, 万物复苏 (The spring breeze blows the land, reviving everything).

树根支撑枝叶,花繁叶茂 (The root supports the crown, growing the forrest).

The two parallel sentences don't share any word so that the exact match strategy fails to identify the alignment between them. However, the alignment can be captured based on POS, syntactic role and semantic matches.

3.2 Sequence Alignment Measures

We use the following algorithms to measure the alignment between sentences.

Longest Common Subsequence(LCSeq) Longest Common Subsequence algorithm is a commonly used approach to compare multiple sequences (Hirschberg, 1977). The subsequences are not required to occupy consecutive positions within the original sequences. Parallel sentences often contain longer common subsequences. The LCSeq algorithm can be effectively solved by using dynamic programming. Given two sequence $X = (x_1, x_2, ..., x_m)$ and $Y = (y_1, y_2, ..., y_n)$, the prefixes of X are X_i, i from 1 to m; the prefixes of Y are Y_j, j from 1 to n. Let $LCSeq(X_i, Y_j)$ represent the set of longest common subsequence of prefixes X_i and Y_j. This set of sequences can be got in the way below.

$$LCSeq(X_i, Y_j) = \begin{cases} \oslash & \text{if } i = 0 \text{ or } j = 0 \\ LCSeq(X_{i-1}, Y_{j-1}) \cup x_i & \text{if } R(x_i, y_j) = 1 \\ longest(LCSeq(X_i, Y_{j-1}), LCSeq(X_{i-1}, Y_j)) & \text{if } R(x_i, y_j) = 0 \end{cases}$$

$R(x_i, y_j)$ represents the condition of word alignment and can be realized using strategies in §3.1. We compute the LCSeq and the normalized length of LCSeq (NormLCSeq) for sentences s_i and s_j as features. NormLCSeq is computed as Equation 1.

$$NormLCSeq(s_i, s_j) = \frac{|LCSeq(s_i, s_j)|}{max(|s_i|, |s_j|)} \tag{1}$$

Longest Common Substrings (LCStr) Parallel sentences have a high chance to have common substrings. Therefore, we compute the longest common substrings of two sentences. Different from LCSeq, the common substrings are required to occupy consecutive positions. Therefore, high LCStr indicates a better local alignment. Different word alignment strategies can be applied. We use the length of the longest common substring as a feature.

Needleman-Wunsch Algorithm(NW) We adapt the Needleman-Wunsch algorithm (Needleman and Wunsch, 1970) for our task. This algorithm is widely used in computational biology for finding sequence alignment among genes. Compared with LCS, it looks for an alignment between whole sequences, which maximizes an overall score function as the sum of the scores over all aligned element pairs in two sequences.

Given two sequences $X \in \mathcal{V}^*$ and $Y \in \mathcal{V}^*$, an alignment can be represented as a two-dimensional array $Align_{2\times I}^{X,Y}$ that every word in one sequence is aligned to one word in the other sequence or to an indel which is caused by inserting a word into one sequence or deleting a word from the other sequence, where I is the number of aligned element pairs. The alignment score is computed as Equation 2.

$$AlignScore(X, Y) = \sum_{i=1}^{I} S(Align_{0,i}^{X,Y}, Align_{1,i}^{X,Y}) \tag{2}$$

where $S(x, y)$ assigns a score between a pair of aligned elements.

To compute $S(x, y)$, there are two types of parameters: a *gap penalty* and a *substitution matrix*. Gap penalty values are used to penalize the score when a word in one sequence is aligned to an indel in the other. Therefore,

NW algorithm would penalize the long distance matches. The substitution matrix is used to assign alignment scores between every pairs of words. A good pair alignment will be rewarded with a higher score. Obviously, the word alignment strategies we discuss in §3.1 can be used here to construct the substitution matrix.

The alignment algorithm runs based on dynamic programming. Please refer to the details in (Needleman and Wunsch, 1970). Once we get the best alignment, we can get a best $AlignScore$. This score is correlated to the length of sequences. To reduce this effect, we use the normalized score, as shown in Equation 3, to build features .

$$NormAlignScore(X, Y) = \frac{AlignScore(X, Y)}{I} \tag{3}$$

3.3 Tree Alignment Measures

We also exploit syntactic structures. Tree kernels are the natural way to exploit syntactic structural properties, which compute the similarities between parsed trees without enumerating the whole fragment space. In this work, we parse sentences with a dependency parser. We use the Partial Tree (PT) kernel (Moschitti, 2006) to measure the similarity between two trees, since it is suitable for dependency parsing. In addition, partial tree kernel considers the ordered child sequence, which makes it suitable for our task as well.

The PT kernel is defined as:

$$K(T_1, T_2) = \sum_{n_1 \in N_{T_1}} \sum_{n_2 \in N_{T_2}} \triangle(n_1, n_2) \tag{4}$$

where N_{T_1} and N_{T_2} are the sets of nodes in T_1 and T_2, respectively and $\triangle(n_1, n_2)$ indicates the number of common fragments rooted at the n_1 and n_2 nodes. The kernels can be effectively computed based on dynamic programming (Moschitti, 2006):

- if $R(n_1, n_2) = 0$, then $\triangle(n_1, n_2) = 0$
- else $\triangle(n_1, n_2)$ would be computed recursively on the sets of ordered child sequences of n_1 and n_2.

Again, we can utilize strategies introduced in §3.1 to compute $R(n_1, n_2)$. Figure 1 shows a dependency parsing tree of an example sentence. We use the normalized kernel values as features, $K^{norm}(T_1, T_2) = \frac{K(T_1,T_2)}{\sqrt{K(T_1,T_1)K(T_2,T_2)}}$.

3.4 Location and Length Features

We observe that parallel sentences also locate regularly in discourse. For example, they usually occupy consecutively within the same paragraph, or locate symmetrically in multiple paragraphs. In addition, they often have close length and close number of clauses. We use the following features to describe these observations.

- **Adjacency**: if two sentences in the same paragraph, and the absolute difference of sentence indexes is smaller than 3, the feature value is set to 1, otherwise it is set to 0.
- **Location Alignment**: This feature is based on the sentence positions. If two sentences are in different paragraphs and they are both the first sentence in the paragraphs, or both the last sentence in the paragraphs, the feature value is set to 1, otherwise it is set to 0.
- **Length difference**: The absolute length difference of two sentences.
- **Clause difference**: If the number of clauses is the same, the feature is set to 1, otherwise it is set to 0. The clauses are segmented by commas.

3.5 Summarization of Features

We summarize the used features in Table 2. Except for location and length features, we use various alignment measures together with different word alignment strategies to generate features. We use both the absolute and normalized length of LCSeq scores as features. LCStr focuses on local consecutive matches, therefore we only use the length of longest common substrings as features. The tree kernel method deals with syntactic structural properties, therefore we construct two trees for each sentence based on dependency parsing. We use POS tags and dependency tags respectively as the values of the tree nodes.

Feature Set	Feature	Word Alignment Strategy
LCSeq	$\|LCSeq_{exact}(s_i, s_j)\|$, $NormLCSeq_{exact}(s_i, s_j)$	Exact match
	$\|LCSeq_{pos}(s_i, s_j)\|$, $NormLCSeq_{pos}(s_i, s_j)$	POS match
	$\|LCSeq_{semantic}(s_i, s_j)\|$, $NormLCSeq_{semantic}(s_i, s_j)$	Semantic match
	$\|LCSeq_{syntacticrole}(s_i, s_j)\|$, $NormLCSeq_{syntacticrole}(s_i, s_j)$	Syntactic role match
LCStr	$\|LCStr_{exact}(s_i, s_j)\|$	Exact match
	$\|LCStr_{pos}(s_i, s_j)\|$	POS match
	$\|LCStr_{semantic}(s_i, s_j)\|$	Semantic match
	$\|LCStr_{syntacticrole}(s_i, s_j)\|$	Syntactic role match
NW	$NormAlignScore_{exact}(s_i, s_j)$	Exact match
	$NormAlignScore_{pos}(s_i, s_j)$	POS match
	$NormAlignScore_{semantic}(s_i, s_j)$	Semantic match
	$NormAlignScore_{syntacticrole}(s_i, s_j)$	Syntactic role match
Tree Alignment	$K^{norm}_{pos}(s_i, s_j)$	POS match
	$K^{norm}_{syntacticrole}(s_i, s_j)$	Syntactic role match
LocLen	Adjacency	—
	Location Alignment	—
	Length Difference	—
	Clause Difference	—

Table 2: Summarization of the features for a sentence pair $< s_i, s_j >$.

3.6 Parallelism Chunk Identification

Given an essay, once every pair of sentences is classified as *parallel* or *non-parallel*, we construct parallelism chunks based on the classification results. We use an aggressive strategy based on transitivity: if two sentence pairs $< x, y >$ and $< x, z >$ are parallel, then $< x, y, z >$ forms a parallelism chunk, no matter whether pair $< y, z >$ is classified as parallel.

4 Evaluation

4.1 Experimental Settings

Data and Classifiers We split the essays in our dataset into 5 parts and run cross-validation. Each time, 4 parts are used for training and the remaining part is used for test. Sentences from the same parallelism chunks form a set of positive pairs, while sentences that are in the same essay but not parallel form negative pairs.

The word embeddings for semantic similarity computation are learned using the Word2Vec tool (Mikolov et al., 2013) on a dataset consisting of 85,000 student essays collected from the web. The dimension of word embeddings is 50. The size of vocabulary is about 490,000.

We adopt the Random Forests (Breiman, 2001) as the classifier and use the implementation in Scikit-learn toolkit (Pedregosa et al., 2011) with default parameters.

Evaluation Metrics We adopt precision, recall and F_1 score as evaluation metrics. The metrics can be computed at pair-wise level and parallelism chunk level respectively.

At pair-wise level, the precision and recall are computed as:

$$\textit{pair-precision} = \frac{\#\text{correctly identified parallelism sentence pairs}}{\#\text{identified parallelism sentence pairs}} \tag{5}$$

$$\textit{pair-recall} = \frac{\#\text{correctly identified parallelism sentence pairs}}{\#\text{true parallelism sentence pairs}} \tag{6}$$

At chunk level, the precision and recall are computed as:

$$\textit{chunk-precision} = \frac{\#\text{correctly identified parallelism chunks}}{\#\text{identified parallelism chunks}} \tag{7}$$

$$\textit{chunk-recall} = \frac{\#\text{correctly identified parallelism chunks}}{\#\text{true parallelism chunks}} \tag{8}$$

A correctly identified parallelism chunk means the identified chunk has the same sentences with a labeled chunk. In both cases, $F_1 = \frac{2 \times precision \times recall}{precision + recall}$.

Feature Set	pair-P	pair-R	pair-F_1	chunk-P	chunk-R	chunk-F_1
LocLen	0.71	0.41	0.52	0.38	0.32	0.35
LCSeq	0.72	0.64	0.68	0.45	0.49	0.47
LCStr	0.71	0.63	0.67	0.46	0.48	0.47
NW	0.75	0.68	0.72	0.49	0.55	0.52
Tree alignment	0.67	0.44	0.53	0.33	0.31	0.32
NW + LocLen	**0.85**	0.78	0.81	0.68	0.70	0.69
NW + Tree alignment	0.81	0.70	0.75	0.58	0.59	0.59
ALL	**0.85**	**0.79**	**0.82**	**0.73**	**0.70**	**0.72**

Table 3: Evaluation results of using different alignment measures.

Word Alignment Strategy	pair-P	pair-R	pair-F_1	chunk-P	chunk-R	chunk-F_1
Exact	0.72	0.69	0.71	0.47	0.54	0.50
POS	0.49	0.51	0.50	0.23	0.31	0.26
Syntactic role	0.77	0.59	0.67	0.52	0.52	0.52
Semantic	0.79	0.66	0.72	0.56	0.57	0.56
Exact + Syntactic role	0.81	0.71	0.76	0.62	0.65	0.64
Exact + Semantic + Syntactic role	0.82	0.72	0.77	0.64	0.65	0.64
Exact + Semantic + Syntactic role + POS	0.81	0.72	0.76	0.62	0.65	0.63

Table 4: Evaluation results of using different word alignment strategies.

4.2 Results

Table 3 shows the experimental results using different sequence alignment measures. All word alignment strategies are used in this experiment. The best alignment measure is the score computed using the Needleman-Wunsch algorithm. This is reasonable, since it captures the alignment on the whole sequence, considering the local alignments like LCStr and penalizing long distance matches, which LCSeq ignores.

Different from LCStr, LCSeq and NW, tree alignment exploits more complex structural information. So tree alignment based measures should complement sequence based measures. The best combination of tree based and sequence based measures are NW plus tree alignment.

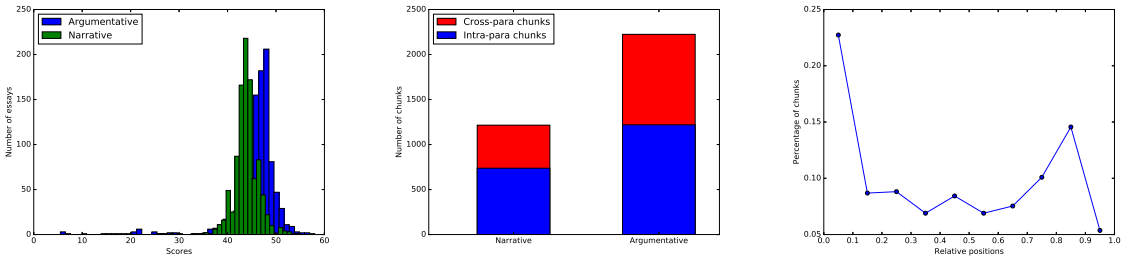
Using the location and length features (LocLen) alone leads to a low recall, but they can improve the performance when combining with other features. We can see that combining LocLen and NW achives good performance.

Combining all alignment measures achieves the best performance. The results demonstrate that pair-wise sentence parallelism can be effectively identified. The chunk-wise performance is moderate. The precision of pair-wise classification is shown to be more crucial to the chunk-wise performance.

Feature	Weight
$NormAlignScore_{semantic}$	0.161
$NormAlignScore_{exact}$	0.148
$NormAlignScore_{syntacticrole}$	0.117
$NormAlignScore_{pos}$	0.105
Location Alignment	0.078
$K^{norm}_{syntacticrole}$	0.062
$NormLCSeq_{exact}$	0.056
Length Difference	0.047
K^{norm}_{pos}	0.040
$LCStr_{exact}$	0.036

Table 5: Top ranked feature weights.

We are also interested in the contributions of various word alignment strategies. Table 4 shows the performance of using different word alignment strategies and their combinations. All alignment measures are used in this experiment. We can see that the semantic match strategy performs best. This indicates that semantic level information is important. We observe that the best combination of two strategies is the combination of exact match and syntactic role match, while the best combination of three strategies is the combination of exact match,



(a) Distribution of essays scores.

(b) Number of chunks in narrative and argumentative essays.

(c) Distribution of chunks' relative positions.

Figure 2: Basic statistics of the student essay dataset and the use of sentence parallelism in the dataset.

semantic match and syntactic role match. Different strategies complement each other except that the POS match doesn't provide extra gain in our experiments. Table 5 shows the feature weights learned by the Random Forests model. The trend is similar to the previous observations.

5 Sentence Parallelism and Essay Writing

The automated sentence parallelism identification makes it possible to study the use of sentence parallelism in student essays and its relation to essay quality on large datasets. We collected another dataset containing essays written by senior high school students in mock examinations. This dataset has 1036 narrative essays and 1064 argumentative essays, and it doesn't overlap with the dataset introduced in §2. All these essays had been scored by professional high school teachers. The scores ranges from 0 to 60. The distribution of essay scores is shown in Figure 2(a). We can see the distribution of either narrative or argumentative essays meets the normal distribution. The dataset should be representative to reflect the real situation.

5.1 How Students Use Sentence Parallelism

We use our system to process these essays and extract parallelism chunks. We extract 2224 and 1219 parallelism chunks in argumentative essays and narrative essays respectively. These parallelism chunks can be categorized into two types: *intra-para chunks* and *cross-para chunks*. Intra-para chunks contain parallel sentences within the same paragraph, while cross-para chunks have parallel sentences across multiple paragraphs.

The ratio of each type of chunks are shown in Figure 2(b). Parallelism chunks, especially the ones that cross paragraphs, are used more often in argumentative essays than in narrative essays. We examine some essays and find that in argumentative essays, students would use parallel sentences to express their main ideas that are used to support the thesis from different aspects. The parallelism can add the clarity in organization. Therefore, there are more cross-para chunks in argumentative essays.

We also examine the relative positions of parallelism chunks. We use the average sentence number of sentences in a chunk divided by the number of sentences in the essay as the relative position of the chunk. Figure 2(c) shows the distribution of relative positions, which are grouped into 10 zones. We can see the distribution is interesting that sentence parallelism is used much more often in the beginning or the ending of essays, while relatively less parallel sentences are used in the body part. We guess the reason is that since parallel sentences are used to impress the readers, the beginning and the ending parts are easier to draw readers' attentions. As a result, students tend to put impressive sentences at these important positions.

Essay type	#intra-para chunks	#cross-para chunks	#all chunks	Presence
Narrative	0.146	0.082	0.161	0.146
Argumentative	0.20	0.233	0.290	0.299

Table 6: Pearson coefficients between scores and the number of different types of parallelism chunks.

5.2 Sentence Parallelism and Essay Scores

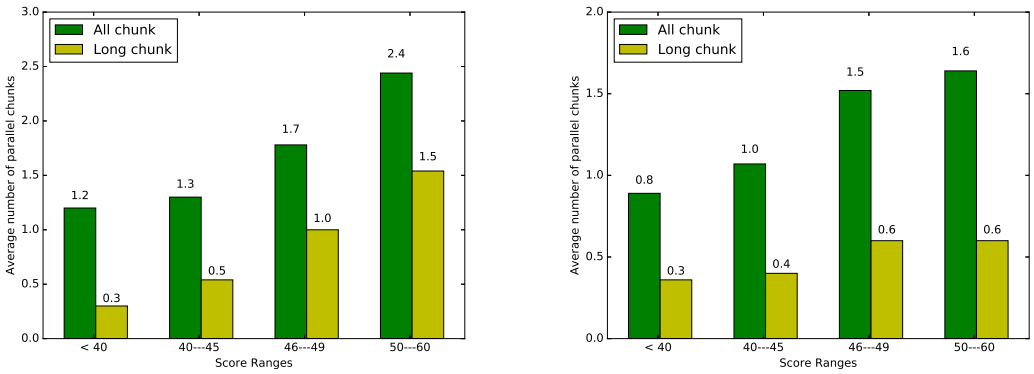
Does the use of sentence parallelism relate to the quality of essays? To answer this question, we analyze the relationship between sentence parallelism and essay scores. We first compute the pearson correlation coefficients between the number of different types of parallelism chunks and the scores of essays. Table 6 shows the results. The correlation coefficient with the number of chunks can reach 0.29 in argumentative essays. In contrast, the

Essay type	All essays	Presence	Absence
Narrative	43.74	44.09	43.13
Argumentative	45.65	46.35	42.13

Table 7: Average scores of essays in terms of the presence or absence of sentence parallelism.

correlation in narrative essays is much lower. The number of cross-para chunks has a higher correlation to essay scores in argumentative essays, but a lower correlation in narrative essays, compared with the number of intra-para chunks.

If we consider the presence or absence of parallelism, the correlations have the same trends as shown in the last column in Table 6. Table 7 shows the average scores of essays in terms of the presence or absence of sentence parallelism. In both narrative and argumentative essays, essays with sentence parallelism tend to have higher scores in average compared with the ones without sentence parallelism. Both differences are significant at 95% level (t-test, $p < 0.05$). The score difference is more obvious in argumentative essays.



(a) Argumentative essays.

(b) Narrative essays.

Figure 3: Average number of parallelism chunks in essays coming from different score ranges.

We further examine sentence parallelism in essays within 4 different score ranges, including 40 points below, 40 to 45, 46-49 and 50-60. We compute the average number of parallel sentence chunks in each score range. The result is shown in Figure 3 for argumentative and narrative essays respectively. We can see that the essays from higher score ranges have a larger average number of parallelism chunks. The trend holds for both argumentative and narrative essays.

We also want to consider the effect of the quality of sentence parallelism. Instead of dealing with the content, we consider *long chunks* whose chunk sizes are equal to or greater than 3. We simply view long chunks as high quality parallelism. The results are also shown in Figure 3. The differences among score ranges in argumentative essays are more obvious so that high quality sentence parallelism might be a useful indicator of well written argumentative essays. In contrast, long chunks appear much less in narrative essays across all score ranges.

5.3 Discussion

This section have studied the relationship between the use of sentence parallelism and the types and quality of student essays. The biggest observation is that the essay type—argumentative or narrative essay—is a key factor when we study sentence parallelism. First, the frequency and styles of using sentence parallelism are different. Parallelism is much more often used in argumentative essays. The ratio of cross-para chunks is also higher in argumentative essays. Second, the frequency or presence of using sentence parallelism has a positive correlation to the quality of essays. The correlation exists but is weak in narrative essays, while it is stronger in argumentative essays. According to the score distributions, recognizing high quality and low quality essays is crucial and challenging. High quality sentence parallelism may be useful to distinguish good and poor argumentative essays.

Notice that the ways of using parallelism may relate to how students are taught on writing, which might be different across countries and cultures. The observed statistics may also be affected by the topics of essays. Nonetheless, the observations should potentially help to design features for automated writing evaluation.

6 Related Work

6.1 Sequence Alignment

Finding the common parts among sequences have been a set of classic computer science problems. The typical problems include finding the longest common subsequence (Hirschberg, 1977), longest common substring and multiple sequence alignment (Carrillo and Lipman, 1988; Needleman and Wunsch, 1970). These techniques are commonly used in computational biology and also applied to natural language processing for constructing concept mapping dictionary (Barzilay and Lee, 2002), identifying sentence level paraphrases (Barzilay and Lee, 2003) and modeling the organization of student essays (Persing et al., 2010). In this work, we exploit these alignment measures for deriving features, since parallel sentences should have a kind of alignment.

6.2 Semantic Similarity of Texts

A large of body of previous work focuses on measuring the semantic similarity of texts. Semantic similarity of text usually depends on exploiting the semantic similarity of words and concepts (Corley and Mihalcea, 2005; Mitchell and Lapata, 2008). While the semantic similarity of words and concepts are learned based on distributional statistics (Lin, 1998; Padó and Lapata, 2007). Recently, neural networks based methods are proposed to learn the distributed representation of words on large scale of corpus (Mikolov et al., 2013). The learned word embeddings enable similar words to have a close distance in the vector space. There is also work on sentential paraphrase identification (Madnani and Dorr, 2010). Paraphrases are different expressions that convey the same meaning. Although it is similar to our task, the goals are different, since parallel sentences are not expected to have the same meaning and paraphrases are not required to have similar structures. Many researchers also exploit the structural properties of sentences to measure semantic similarity of texts, such as the tree kernel emthods (Moschitti, 2006; Mooney and Bunescu, 2005; Culotta and Sorensen, 2004).

6.3 Text Quality Analysis

Some work focuses on dealing with rhetorical device such as recognizing metaphor in texts (Shutova, 2010). Parallelism is also an important rhetorical device. Hobbs and Kehler (1997) study the clause level parallelism. However, little work has been done on sentence-level parallelism identification. These is work on predicting the quality of articles (Louis and Nenkova, 2013; Pitler and Nenkova, 2008), writing styles (Ashok et al., 2013) and student essays (Attali and Burstein, 2006). They mainly use simple shallow features, but seldomly use rhetorical device related features. Automated rhetorical device analysis should help to improve the above tasks.

7 Conclusion

We have investigated identifying sentence parallelism in student essays. We adopt machine learning to learn a prediction model based on an annotated dataset. We study various alignment measures and different word alignment strategies for deriving features. The evaluation demonstrates that our proposed method can effectively identify sentence parallelism, achieving a F_1 score of 82% at pair-wise level and 72% at parallelism chunk level. We also study the use of sentence parallelism in more than 2000 student essays based on automated parallelism identification. We find that students tend to use more sentence parallelism in argumentative essays compared with narrative essays. The essays with sentence parallelism have higher scores in average. The presence of high quality sentence parallelism shows to be an indicator of high quality argumentative essays.

Acknowledgements

This research is supported by the National Natural Science Foundation of China (No.61402304,No.61303105), the Beijing Municipal Natural Science Foundation (No.4154065) and the Humanity & Social Science General Project of Ministry of Education (No.14YJAZH046).

References

Vikas Ganjigunte Ashok, Song Feng, and Yejin Choi. 2013. Success with style: Using writing style to predict the success of novels. *Poetry*, 580(9):70.

Yigal Attali and Jill Burstein. 2006. Automated essay scoring with e-rater® v. 2. *The Journal of Technology, Learning and Assessment*, 4(3).

Regina Barzilay and Lillian Lee. 2002. Bootstrapping lexical choice via multiple-sequence alignment. In *EMNLP 2002*, pages 164–171. Association for Computational Linguistics.

Regina Barzilay and Lillian Lee. 2003. Learning to paraphrase: An unsupervised approach using multiple-sequence alignment. In *NAACL 2003 - Volume 1*, pages 16–23. Association for Computational Linguistics.

Leo Breiman. 2001. Random forests. *Machine learning*, 45(1):5–32.

Jean Carletta. 1996. Assessing agreement on classification tasks: the kappa statistic. *Computational linguistics*, 22(2):249–254.

Humberto Carrillo and David Lipman. 1988. The multiple sequence alignment problem in biology. *SIAM Journal on Applied Mathematics*, 48(5):1073–1082.

Wanxiang Che, Zhenghua Li, and Ting Liu. 2010. Ltp: A chinese language technology platform. In *Proceedings of the 23rd International Conference on Computational Linguistics: Demonstrations*, pages 13–16. Association for Computational Linguistics.

Courtney Corley and Rada Mihalcea. 2005. Measuring the semantic similarity of texts. In *Proceedings of the ACL workshop on empirical modeling of semantic equivalence and entailment*, pages 13–18. Association for Computational Linguistics.

Aron Culotta and Jeffrey Sorensen. 2004. Dependency tree kernels for relation extraction. In *Proceedings of the 42nd Annual Meeting on Association for Computational Linguistics*, page 423. Association for Computational Linguistics.

Daniel S Hirschberg. 1977. Algorithms for the longest common subsequence problem. *Journal of the ACM (JACM)*, 24(4):664–675.

Jerry R Hobbs and Andrew Kehler. 1997. A theory of parallelism and the case of vp ellipsis. In *Proceedings of the eighth conference on European chapter of the Association for Computational Linguistics*, pages 394–401. Association for Computational Linguistics.

Dekang Lin. 1998. An information-theoretic definition of similarity. In *ICML*, volume 98, pages 296–304. Citeseer.

Annie Louis and Ani Nenkova. 2013. What makes writing great? first experiments on article quality prediction in the science journalism domain. *Transactions of the Association for Computational Linguistics*, 1:341–352.

Nitin Madnani and Bonnie J Dorr. 2010. Generating phrasal and sentential paraphrases: A survey of data-driven methods. *Computational Linguistics*, 36(3):341–387.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119.

Jeff Mitchell and Mirella Lapata. 2008. Vector-based models of semantic composition. In *ACL*, pages 236–244.

Raymond J Mooney and Razvan C Bunescu. 2005. Subsequence kernels for relation extraction. In *Advances in neural information processing systems*, pages 171–178.

Alessandro Moschitti. 2006. Efficient convolution kernels for dependency and constituent syntactic trees. In *European Conference on Machine Learning*, pages 318–329. Springer.

Saul B Needleman and Christian D Wunsch. 1970. A general method applicable to the search for similarities in the amino acid sequence of two proteins. *Journal of molecular biology*, 48(3):443–453.

Sebastian Padó and Mirella Lapata. 2007. Dependency-based construction of semantic space models. *Computational Linguistics*, 33(2):161–199.

F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. 2011. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830.

Isaac Persing, Alan Davis, and Vincent Ng. 2010. Modeling organization in student essays. In *EMNLP 2010*, pages 229–239. Association for Computational Linguistics.

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

Ekaterina Shutova. 2010. Models of metaphor in nlp. In *Proceedings of the 48th annual meeting of the association for computational linguistics*, pages 688–697. Association for Computational Linguistics.