A Machine Learning Approach to Identify Textual Parallel Structure

Jingying Wang

Abstract

Parallel structure is a ubiquitous phenomenon in different languages. The application of this structure could affect cognitive processing as well as emotional response. As parallelism is also one of the important rhetoric devices, this structure is worth corpus-based quantitative studies and should be taken seriously in machine translation and some other NLP tasks. A fundamental step for further studies might be to detect this structure automatically. Parallelism lies in diverse registers and in different levels of a text. In this paper, the focus is the syntactic parallelism at inner-sentence level and a supervised machine learning approach to identify such parallel structures will be proposed. The experimental dataset for this project is American Inaugural Speech Corpus which contains both automatic PoS-tagging and manual annotation. The feature engineering process, especially how to convert linguistic features into computational features in a relatively balanced way, are also explained in this paper. Then the detecting performance of classifiers trained with different machine learning algorithms and with different features combinations will be critically evaluated coupled with error analysis. The classifier developed in this project shows good performance. Further work on automatic parallelism identification and possible application scenarios will be discussed at the end of this article.

1 Introduction

1.1 Motivation

In linguistics, parallelism, also known as parallel structure or parallel construction, usually refers to a structure within one or more sentences which contains elements (phrases or clauses) that have similar textual pattern and in the same semantic level. This is structure can be found across languages and in diverse registers ranges from political speeches, poetries to daily conversations. Parallelism can be constructed in different textual levels: phonologic level (e.g. alliteration), morphologic level (e.g. identical endings, anaphora), syntactic level, semantic level etc. (Winfried Menninghaus et al., 2017) and one sentence could contain multiple layers of parallelism. There are experimental studies suggest that parallelism with or without coordinated structures can reduce memory demands, speed up cognitive processing and thus facilitate sentence comprehension (Lyn Frazier et al. 1984; Patrick Sturt et al. 2010). Parallelism as a rhetoric device has powerful effects on emotional responses (Winfried Menninghaus et al., 2017) and can generate some sense of rhythm (Song et at., 2016), and therefore is very commonly used in poetries and public speeches. The previous studies about parallelism are mostly qualitative studies. As it is an important textual structure, parallelism is also worth studying quantitatively, with corpus-linguistic methods, for example. Additionally, this structure should also be taken seriously in machine translation, text generation, text evaluation and some other NLP tasks, especially when it is as a rhetoric device. For all these kinds of further studies, the fundamental step should be to identify this structure automatically.

1.2 Related Work

Studies on automatic textual parallelism identification are still very few (Mu et al., 2018) and studies specifically on automatic identification of English parallelism are even fewer. In fact, most of these kinds of studies are done by Chinese researchers and specifically for Chinese parallel

sentence/sentence pair (block) extraction. Different approaches and techniques have been tried for this kind of task. Some projects developed rule-based extraction systems (熊李艳 et al., 2018; Liang et al., 2013), some applied machine learning approaches (Song et at., 2016), some applied supervised deep learning approaches based on convolutional neural network (穆婉青 et at., 2018) or recurrent neural network (Dai et at., 2018) and some used unsupervised clustering (Guégan and Hernandez, 2006). Features taken into account are usually lexical features (e.g. word recurrence), syntactic/structural features and semantic features. These features are implemented in different ways with different algorithms. For instance, structural similarity is computed based on Charniak's parser together with Wagner & Fischer's string edit distance algorithm (Guégan and Hernandez, 2006) or based on PoS tags with Longest Common Subsequence (LCS) algorithm or Needleman-Wunsch Algorithm (NW) (Song et at., 2016). Semantic features are represented by word vectorization (Dai et at., 2018; 穆婉青 et at., 2018; Song et at., 2016) or by semantic standardization with WordNet (Guégan and Hernandez, 2006).

The performances of these different approaches are hard to compare. The reasons are as follows:

(1) Datasets are different: The types of corpus used as experimental data are quite divers: Literatures written in classic Chinese during Pre-Qin times (Liang et al., 2013), Student essays (Dai et at., 2018; Song et at., 2016), proses from textbooks or exams for senior high school (穆婉青 et at., 2018), political reports or speeches (熊李艳 et al., 2018) or scientific articles (Guégan and Hernandez, 2006) are employed for different projects.

(2) Definition of parallelism varies across projects and across language: Most of these studies focus on cross-sentence or cross-paragraph level of parallelism. Some focuses on inner-sentence level. Definition of "sentence" is also not exactly the same due to different treatment of semicolon. Furthermore, there are also some differences between Chinese parallelism and English parallelism, e.g. while Chinese parallelism usually consists of three or more similar elements, English parallelism normally can be formed with only two parallel elements; While English parallelism is often formed by conjunctions such as "and", "or", "nor" etc., Chinese parallelism depends much less on conjunctions (张伯菁, 2002).

(3) Quality of manual annotation work might be different: Since the definition of parallelism is not the same in different projects, the tagset and guidelines are also slightly different for different manual annotation task. Additionally, most of these papers did not mention how the quality of manual annotation work (Inter-Annotator Agreement) is measured. Nevertheless, some systems for Chinese parallelism identification/extraction have shown satisfying performances (熊李艳 et al., 2018; Dai et at., 2018; 穆婉青 et at., 2018; Song et at., 2016). In comparison, the existing system for English parallelism (inter-sentence level) identification although outperforms the baseline set by the authors themselves, the F-measure of which is still very low (best overall F-measure is 54%) (Guégan and Hernandez, 2006). It is clear that more research on this field (English) is needed.

1.3 Overview of This Project

In this paper, a supervised machine learning approach for English parallelism identification based on LCS algorithm, coordinator[1] locating and some other feature extraction methods will be proposed. The main steps of this project are shown in Fig. 1:

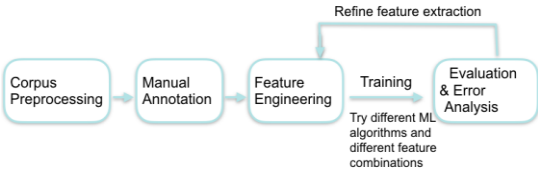

Fig. 1: Workflow of the project

The focus of this project is the parallelism at syntactic level, more specifically, inner-sentence level. The following section (Section 2) is about dataset: How the experimental data is established based on American Inaugural Speech Corpus. Corpus preprocessing and manual annotation work are also discussed in this section. Section 3 is about classifier training and evaluation: Which patterning features could be informative for identifying parallelism and how these linguistic

[1] Coordinator here refers to normalized coordinating structure which will be explained later

features are interpreted in programming language (Python). The performance of the classifiers trained with different machine learning algorithms and with different features combinations will be critically evaluated. Strategies for refining feature extraction will be explained based on error analysis. The last part (Section 4) gives insights about application possibilities of such classifier and future work.

2 Dataset

2.1 Employed Corpus

American Inaugural Speech Corpus is one of the most frequently used English corpus and is of high quality. Parallelism is often applied for rhetoric purpose in political speeches and the language use in this corpus is formal, so that this corpus should be suitable for this project. 3 inaugural speeches respectively from 2001, 2009 and 2017 by three different US presidents with altogether 5741 tokens (380 sentences) are chosen to be processed. The size of this dataset is definitely small, but for illustration purpose, this size could be accepted.

2.2 Preprocessing

a) PoS-Tagging: Structural similarity is the core feature of parallel sentence. This feature can be revealed by Part-of-Speech tag sequences, more specifically, 2 or more identical or similar PoS-tag sequences are contained within one sentence. Due to the fact that the state-of-the-art PoS-tagger has not reached 100% accuracy and sometimes word class assignment can be ambiguous even for human, tag sequences generated by automatic PoS-tagging system might be different from what we expected. In order to observe the behavior of the tagger and make the best use of it to interpret syntactic features, texts are first of all PoS-tagged (Format: *WORD_TAG*) before manual annotation. For this task, Stanford POS Tagger (Version: 3.9.2 2018-10-16)[2] is used. Despite the fact that this preprocessing method added difficulty to reading, it actually facilitates feature extraction process as well as error analysis.

b) Segmentation: The border of the sentence can sometimes be hard to define, especially when it comes to the usage of semicolon. They are often described as being more powerful than commas, while not quite as a strong as periods (full stops)[3]. After analyzing the role semicolon plays in different texts, it is found that, the text blocks separated by semicolons are usually complete sentences rather than phrases. In fact, if semicolons are not treated as sentence closer, the American Inaugural Speech Corpus can provide us sentences which contain more than 200 words. In this case, even if the classifier correctly identifies that there is a parallel structure embedded in this "sentence", it is still inefficient for people to recognize the specific part from an extremely long sentence. So for this project, the chosen sentence closers are dot (full stop), exclamation mark, question mark, colon and semicolon (. ! ? : ;).

2.3 Manual Annotation

The workflow for this task is as follows: 1). First annotator annotates one speech with starting guidelines and refine the guidelines during this process. 2). Second annotator annotates the same text with the revised guidelines independently. 3). Two annotators discuss about sentences received different tags to reach consensus and refine guidelines together. 4). Both annotators finish annotation work independently for the rest 2 texts and then repeat step 3 to assure the quality of the data, Kappa value is computed after step 2 and 4 for every text to measure inter-annotator agreement.

2.3.1 Tagset and Guidelines

Annotation Guidelines (simplified version[4]):

(1) The following 4 types of parallelism (including but not limited to) are tagged as "t":

Note: "**cc**" refers to coordinating words, namely "and", "or", "but", "while (as in conj.)" or part of a paired conjunction, e.g. "not only… but also…"; "either… or…"; "**A, B, C**" refer to the parallel elements with similar grammatical structures.

type 1: …A cc B…e.g. Yet every so often, the oath is taken amidst **gathering clouds** and **raging storms**. (Obama, 2009)

type 2: …A, B… e.g. Homes have been lost, **jobs shed, businesses shuttered**. (Obama, 2009)

type 3: …A cc B cc C… e.g. Americans are **generous and strong and decent**… (Bush, 2001)

[2] https://nlp.stanford.edu/software/tagger.shtml

3 semicolon. (n.d.) The Farlex Grammar Book. (2016). Retrieved August 11 2019 from https://www.thefreedictionary.com/semicolon

[4] For full version please see Appendix 1.

type 4: …A, B (,) cc C…: e.g. A new national pride will **stir our souls, lift our sights**, and **heal our divisions**. (Trump, 2017)

(2) Sentences with no parallel structure are tagged as "f".

Disagreement during annotation work and the corresponding solution:

It should be noted that, although most people have common sense of what parallel sentences are, the annotation work is not as straightforward as expected. One of the previous studies (Song et at., 2016) mentioned that the cohen's kappa value for their manual annotation is 0.71, which indicates just a moderate level of agreement (McHugh M. L., 2012). And the main disagreement lies in their different judgement standards in terms of the quality of parallelism (Song et at., 2016). This is also the case during this project. Major disagreement occurred when it comes to the following 3 kinds structures:

1) Listing: This raised disagreement about whether or not treating listing of nouns or named entities as parallelism. Because although it does conform with the definition of parallelism that using identical or equivalent syntactic constructions in corresponding clauses or phrases[5], it seems more like an enumeration structure. On the other hand, listing could also have the similar cognitive effect as parallelism. For instance, listing historical locations can intensify readers'/ audience's emotion, so it is decided that this structure should also be identified as parallelism in this project. The linguistic value can be measured based on the context later.

2) Parallel elements with different lengths: Parallelism is not always constructed rigorously with exactly the same sub-sentence structures (Example see below), which makes "similar structure" hard to define. This raised discussion especially about the tolerance of length difference. After analyzing different types of inner-sentence parallelism, we decided to use a tolerance of 2 words. Because this tolerance allows flexible construction to some degree and in the meantime, within this difference, parallel elements can usually be parsed in similar ways (consists of similar constituents) and maintain similar rhythm.

e.g. "But our time **of standing pat, of protecting narrow interests and putting off unpleasant decisions**, that time has surely passed."

3) Partially reversed structure: Conjunction "nor" can help form a partially reversed sentence together with a negation in the previous sub-sentence, e.g. "…**we did not** turn back, **nor did we** falter." After discussion, we agree that this can be seen as a "symmetrical" kind of similarity and should be considered as parallelism, as long as the length difference between 2 elements is within 2 words.

2.3.2 Inter-Annotator Agreement

For quality assurance of manual annotation, Cohen's kappa statistic is used, because it is suitable for measuring inter-annotator agreement between two annotators. Kappa value for 3 annotated texts can be calculated with Scikit-learn library [6]. The following screenshot shows the Kappa score for all texts, which indicates a high level of agreement.

```
Scores:
2001-01-20-George-W-Bush:  0.8113501714998441
2009-01-20-Barack-Obama:  0.8615384615384616
2017-01-20-Donald-J-Trump:  0.8861160625866509
```

Fig. 2 kappa scores for 3 speeches

3 Classifier Training and Evaluation

3.1 Baseline and Start-up features

Baseline: Altogether there are 123 out of 380 sentences tagged as "t" (has parallelism). If all sentences are tagged as "t", the F-measure for "t" is 48.9%, so this score is considered as the baseline for this project.

Feature Extraction (Phase 1):

Start-up features:

(1) Lexical Features:

Feature 1: Number of reoccurred word types. Because parallelism can be formed with the help of repetition of words, e.g. "…stir **our** souls, lift **our** sights, and heal **our** divisions". For better feature extraction, tokens are normalized by lowercasing.

Feature 2: Number of conjunctions. Parallel elements are often coordinated by conjunctions such as "and", "or" etc.

(2) Syntactic Features:

In this project syntactic features are extracted mostly based on automatically assigned PoS-tags.

[5] https://www.thefreedictionary.com/parallelism

[6] https://scikit-learn.org/stable/modules/generated/sklearn.metrics.cohen_kappa_score.html#r219a3b9132e1-1

Feature 3(binary): For type 1 and type 3: Same PoS-tag bigrams/trigrams occur at least 2 times within **one** sub-sentence.

Feature 4(binary): For type 2 and type 4: Same PoS-tag bigrams/trigrams occur in **at least two** sub-sentences.

Feature 5(binary): Similarity of two parallel elements. Sentences like "We_PRP do_VBP not_RB accept_VB this_DT ,_, and_CC we_PRP will_MD not_RB allow_VB it_PRP ._." contain phrases or clauses that have very similar syntactic structure before and after coordinating word/comma. Considering that on the one hand, the tag set is very fine-grained, so some words in similar syntactic and semantic position could be assigned tags which appear to be totally different on the alphabetic surface. For instance, "do_VBP" and "will_MD"; "this_DT" and "it_PRP" these words are assigned different tags. On the other hand, the employed tagger is not 100% accurate, so sometimes PoS tags can be wrongly assigned. Due to these two factors, PoS-tag sequences of sub-sentences can hardly be exactly the same, even when the structures are identical. To tackle this problem and allow certain difference in PoS-tag sequences, Longest Common Subsequence (LCS) algorithm is used. In python, the *difflib* module has implemented this algorithm. The basic idea is to find the longest contiguous matching subsequence that contains no "junk" elements. The same idea is then applied recursively to the pieces of the sequences to the left and to the right of the matching subsequence.[7] In this module, the class SequenceMatcher provides a method *ratio()* that measures the similarity of two sequences. It returns a float in [0, 1], 1 means the sequences are identical and 0 means they have nothing in common. This method is very suitable for this PoS-tag sequence comparing task and we can adjust the threshold for "similar" based on the performance of classifiers.

**** Normalization of coordinating structures***

In order to compare as many coordinated phrase/clause-pairs as possible, normalization of coordinating structures is performed, i.e. conjunction words and commas are treated the same way. However, this could cause some over-normalization problems, because conjunction words can sometimes be part of a parallel element, e.g. "… [honesty_NN and_CC hard_JJ work_NN] ,_, [courage_NN and_CC fair_JJ play_NN]…" In this case, (honesty_NN & hard_JJ work_NN), (hard_JJ work_NN & courage_NN), (courage_NN & fair_JJ play_NN) are compared with each other, and the similarity ratio of these pairs are all below 0.7. Although the similarity ratio between [honesty_NN and_CC hard_JJ work_NN] and [courage_NN and_CC fair_JJ play_NN] is actually 1.0 (exact match), this feature cannot be extracted due to over- normalization. To make sure this kind of structure is also taken into account, the procedure is performed again for the sentence without any normalization, i.e. only compare sub-sentences before and after commas.

**** Why not use syntactic parser to extract syntactic features?***

Although syntactic parser can deliver structural information about the sentence constituents, it is not employed in this project. Reasons are as follows: First of all, syntactic parsing is usually ambiguous especially for long and complex sentences. There could be multiple parsing solutions for one sentence. Even human-beings cannot always be sure about which parsing solution is the right/best one (Frazier, L., 1984), let alone the computers. The example below shows that two PP (prepositional phrase) can be parsed as the second element is subordinated to the first one, even though according to human judgement, they should be in the same syntactic level. (Parser employed: CoreNLPParser[8]. Moreover, the state-of-the-art PoS-tagger has not reached 100% accuracy and the parser strongly depends on PoS-tags, so structural information produced by syntactic parser could introduce more noises and is thus not adopted.

```
(ROOT
  (S
    (NP (PRP it))
    (VP
      (VBZ is)
      (NP
        (NP (DT the) (ADJP (VBN determined)) (NN choice))
        (PP
          (IN of)
          (NP
            (NP (NN trust))
            (PP (IN over) (NP (NN cynicism)))
            (, ,)
            (PP
              (IN of)
              (NP (NP (NN community)) (PP (IN over) (NP (NN chaos))))))))))
    (. .)))
```

Screenshot: Wrongly parsed sentence

3.2 Results (Phase 1)

3.2.1 Parameter settings (1)

[7] See documentation for this module: https://github.com/python/cpython/blob/master/Lib/difflib.py

[8] https://github.com/nltk/nltk/wiki/Stanford-CoreNLP-API-in-NLTK

Optimal Feature-Combination: There are altogether 7 feature extraction methods implemented, namely: F1(number of repeated word types), F2 (number of conjunctions), F3 (inner-subsentence POS-bigram recurrence), F4 (inner-subsentence POS-trigram recurrence), F5 (inter-subsentence POS-bigram recurrence), F6 (inter-subsentence POS-trigram recurrence), F7 (POS-sequence similarity). After experimenting different feature combinations, it is found that for NaiveBayesClassifier (nltk), the optimal features combination is (F1, F2, F3, F5, F7) and for machine learning algorithms from Scikit-learn library, the optimal combination is (F1, F2, F3, F4, F5, F7).

Threshold for Similarity: To select the optimal threshold for Similarity, different similarity ratios are experimented and the corresponding performances of classifiers are observed. After multiple experiments, 0.7 is chosen to be the threshold.

3.2.2 Evaluation Metrics and Cross Validation

Considering that the dataset does not have balanced tags (123 "t" vs. 257 "f"), Accuracy score is not suitable for this evaluation task. Instead, precision, recall and f-measure scores specifically for tag "t" (parallelism) are adopted as evaluation metrics. To tackle the scarce data problem, cross validation with 5 folds is performed for all metrics.

3.2.3 Performance and Error Analysis (1)

Table 1. shows the cross-validated precision, recall and f-measure scores for different classifiers. According to the results, while the classifier based on Naive Bayes algorithm has the highest recall and f-measure scores, the precision of which is nevertheless the lowest. The most contributive features are F1, F2 and F7. The classifier based on LinearSVC (Linear Support Vector Classification) has the highest precision, but the recall and f-measure scores of which are the lowest.

Table 1 Evaluation (1) for tag "t"

Algorithms	P	R	F
NaiveBayesClassifier	0.67	0.93	**0.77**
KNeighborsClassifier	0.73	0.76	0.74
LogisticRegression	0.74	0.69	0.72
LinearSVC	0.76	0.65	0.70

Although every classifier developed in this stage outperforms the baseline, all f-measure scores are below 0.80, which indicates just mediocre prediction. To refine feature extraction and improve the prediction performance, error analysis is necessary.

Typical errors:

A. (Problematic feature extraction: F7) As parallel elements can sometimes be embedded in a much longer sub-sentence, if the whole PoS-tag sequences before and after conjunctions or commas are directly compared, similarity between structure at the same syntactic level cannot be detected. The example below shows that, although "gathering_VBG clouds_NNS" is parallel with "raging_VBG storms_NNS", feature extraction method F7 does not compare the PoS sequence similarity between these two elements, instead the whole sub-sentence before conjunction which consists of 7 PoS-tags and the much shorter PoS sequence (2 tags) are compared. Because of this significant length difference, the similarity ratio is very low and sentence resulted in False Negatives.

*e.g. Yet_CC every_DT so_RB often_RB ,_, **[the_DT oath_NN is_VBZ taken_VBN amidst_IN gathering_VBG clouds_NNS]** and_CC **[raging_VBG storms_NNS]** ._.*

B. (Problematic feature extraction: F3, F4, F5, F6) Sometimes words in totally different syntactic and semantic levels can be assigned exactly same PoS-tag sequences by coincidence. (see example below) As features (inner-subsentence POS-bigram / trigram recurrence) extracted for this kind of sentences are often shared by sentences with parallel structure. Therefore, these sentences can be tagged as "t" and result in False Positives.

*e.g. I_PRP thank_VBP President_NNP Bush_NNP for_IN his_PRP$ service_NN to_TO our_PRP$ Nation_NNP ,_, as_RB well_RB **[as_IN the_DT generosity_NN]** and_CC cooperation_NN he_PRP has_VBZ shown_VBN throughout_**[IN this_DT transition_NN]** ._.*

C. (Problematic feature extraction F1) Long sentences are prone to yield multiple repeated words regardless of parallelism. In the meantime, recurrence of words is one of the most informative features for parallelism. Thus, these long sentences can result in False Positives.

3.3 Refining Feature Extraction (Phase 2)

To address the problems mentioned above, the following strategies are implemented:

• **For error types A: Compare PoS-tag sequences before and after coordinating words/commas in a more controlled way:** After further analyzing characteristics of parallel elements, it is found that, for sentences with parallel structures, parallel elements are not randomly positioned anywhere in a sub-sentence, but usually "glued" together by coordinating words/commas. Additionally, in most cases, at maximum only one of parallel elements is embedded in a longer sub-sentence and the other parallel element(s) take(s) up the whole shorter sub-sentence(s). Therefore, a proper way to extract syntactic feature is to compare POS-tag sequences (contain at least 2 tags) right before and right after the coordinator within similar spans (length difference tolerance corresponds with annotation guidelines: 2 tags). The "base length" should be the length of a shorter sub-sentence.

*** Normalization of PoS-tags**

Since the tag set Stanford POS-Tagger uses (Penn Treebank) is very fine-grained and it differentiate among singular, plural nouns and proper nouns(NN, NNS, NNP(S)); 3rd person and non-3rd person verbs(VBZ, VBP) and so on, this can be disadvantageous for this task, because it degrades the similarity of two parallel elements from the perspective of a computer. For example: "… their_PRP$ pain_NN is_VBZ our_PRP$ pain_NN ,_, their_PRP$ dreams_NNS are_VBP our_PRP$ dreams_NNS…" In this case, although in human eyes, these two parallel elements share the same syntactic structure, the PoS-tag sequences based on Stanford Tagger are different. This is why the threshold for similarity is only 0.7 (see Parameter settings (1)). Low threshold can decrease precision of a classifier significantly, so for better feature extraction, only the first two characters of the PoS-tag are used (except for punctuation-tags which consist of only one character), as they can represent the superordinate word classes. The following pseudocode show how parallel structures between two adjacent sub-sentences is detected by revised feature extraction method:

Pseudocode:

"… not_RB just_RB **on_IN the_DT size_NN of_IN our_PRP$ gross_JJ domestic_JJ product_NN** ,_, but_CC **on_IN the_DT reach_NN of_IN our_PRP$ prosperity_NN** ,_, …"
similar_structure_counter = 0
if the second subsentence is shorter:
    "base length" = length of the second subsentence (6 normalized PoS tags)
    compare "_NN_IN_PR_JJ_JJ_NN" with "_IN_DT_NN_IN_PR_NN" (0.667)
    compare "_DT_NN_IN_PR_JJ_JJ_NN" with "_IN_DT_NN_IN_PR_NN" (0.769)
    compare "_IN_DT_NN_IN_PR_JJ_JJ_NN" with "_IN_DT_NN_IN_PR_NN" (0.857)
    if any of the comparison yield a number bigger than the threshold (0.79)
        similar_structure_counter += 1
……

During this process, similarity between "_IN_DT_NN_IN_PR_JJ_JJ_NN" and "_IN_DT_NN_IN_PR_NN" is detected, which correspond with human judgement. Because of the normalization, threshold for similarity is increased (to 0.79), and therefore the similarity feature in parallelism with partially reversed structure can hardly be extracted through the procedure above. Thus, additional procedure is performed for sentence with the word "nor", in which the reversed sequence of two PoS-tags (without normalization) after "nor" is compared with the PoS-tag sequence for the verb and the preceding subject in the previous sub-sentence. Take "…we did not turn back, nor did we falter." this sentence as an example, if the reversed PoS-tag sequence for "did we" is exactly the same as the PoS-tag sequence for "we did" after "nor", this sentence will also be considered as containing similar structures.

• **For error type B: Ignore long-distance recurrence of PoS-tag bigram/trigram:** For extraction of repetition of PoS-tag bigrams/trigrams, the textual span is also controlled based on coordinating words/commas. That means, if the distance between 2 identical bigrams/trigrams are longer than 2 tokens (coordinating structures usually consist of 2 tokens), the repetition of PoS-tag bigrams/trigrams will not be taken into account.

• **For error type C: Filter out word repetition in "wrong" position:** By analyzing repeated words in parallel elements, it is found that if a word is part of a parallel element, it does not recur randomly anywhere in the sentence, but often in the same position within a certain textual span. For instance, "…, we must [pick ourselves up], [dust ourselves off], …", if the longer sub-sentence is sliced according to the length of a shorter one, then the index of word "ourselves" in the processed sub-sentence is the same as the index of this word in the shorter sub-sentence. Taken this into consideration, the new word-recurrence feature extraction method only detects recurrence in particular positions and thus can filter out some noises from long sentences which often contain multiple repeated words.

3.4 Results (Phase 2)

3.4.1 Parameter settings (2)

Optimal Feature-Combination: Combination of refined feature extraction methods for PoS-tag

sequence similarity, repetition of trigrams and word-recurrence is optimal in this system.

Threshold for similarity: Because of the normalization of PoS-tags in syntactic similarity extraction process, the optimal threshold is increased to 0.79.

3.4.2 Performance and Error Analysis (2)

Table 2 shows that compared with Table 1, f-measure scores of all classifying systems have increased significantly and all above 0.80, which indicates good prediction performance. The system based on LogisticRegression from *Scikit-learn* has the highest precision as well as f-measure. It is worth noting that NaiveBaiyes Classifiers in both phase 1 and phase 2 have the highest recall. If it is desired to identify as many potential parallelisms as possible, then NaiveBaiyes System might be more suitable for the identification task. But in general, classifier based on Logistic Regression model has the best performance.

Table 2 Evaluation (2) for tag "t"

Algorithms	P	R	F
NaiveBayesClassifier	0.8505	0.843	0.841
KNeighborsClassifier	0.8998	0.787	0.833
LogisticRegression	0.8897	0.818	**0.851**
LinearSVC	0.8987	0.818	0.849

Error analysis:

A. Due to errors made by automatic PoS-tagger and ambiguity of word class definition, similar structures might not be identified and result in False Negatives. Take the sentence below as an example, "feed (VB)" is wrongly tagged as NN; "starved" is indeed past participle form of verb "starve", however, in this case it functions as an adjective.

e.g. *"... to_TO [nourish_VB* ***starved_VBN*** *bodies_NNS] and_CC* ***[feed_NN*** *hungry_JJ minds_NNS] ._."*

B. False positives worth a closer look:

e.g. *"They_PRP saw_VBD America_NNP as_IN* ***bigger_JJR than****_IN the_DT sum_NN of_IN our_PRP$ individual_JJ ambitions_NNS ,_ ,* ***greater_JJR than_IN*** *all_PDT the_DT differences_NNS of_IN birth_NN or_CC wealth_NN or_CC faction_NN ._."*

There are some sentences that contain some features of parallelism, but are annotated as "f" according to the strictly defined guideline. Nevertheless, they are identified as containing parallelism by the classifier. This adds some advantages to this classifier, as it can provide us some "borderline" parallelism which is worth analyzing.

4 Discussion

4.1 Possible Application Scenarios

Automatic parallelism identification technique could assist many other tasks and provide fresh perspectives for researchers. Examples of possible applications are as follows:

- Assisting machine translation technique: If parallelism in the source text can be identified and re-constructed in target language, the stylistic feature or rhetoric feature of the source text can be maintained and a higher level of equivalence can be achieved.
- Keyness analysis: As parallelism is often used to strengthen or emphasize the ideas and can intensify the readers' or audiences' emotions, this structure is very suitable to carry ideas which are highly valued by the author (speaker). There are some articles (Al-Ameedi and Mukhef, 2017:195; Almehmdawi 2018:283) show that politicians use this structure to reinforce their values or ideology in their speeches. Therefore, parallelism-sentences could assist researchers in detecting keywords or "aboutness" of a text.
- Text evaluation: As parallelism is one of the important rhetoric devices, it should be taken into consideration when evaluating an essay automatically.

4.2 Future work

As the training data is small, the trained classifier could not generalize ideally on other types of texts, poetries, for example. On the other hand, it might be reasonable to develop different classifiers for specific kinds of texts to serve specific research purposes. In addition, automatic parallelism extraction technique for different textual level, such as cross-sentence or cross-paragraph level is also desired. In order to develop this technique more efficiently and reliably, establishing a shared task with balanced training/test dataset for parallelism identification (preferably for different languages) for competition and comparison is highly recommended.

References

Al-Ameedi, R.T., & Mukhef, R.N. 2017. Aspects of Political Language and Parallelism. Journal of Education and Practice.

Frazier, L., Taft, L., Roeper, T. et al., 1984. Parallel structure: A source of facilitation in sentence comprehension, Memory & Cognition 12: 421. https://doi.org/10.3758/BF03198303

Guégan, Marie & Hernandez, Nicolas. 2006. Recognizing Textual Parallelisms with Edit Distance and Similarity Degree

梁社会,陈小荷,刘浏.先秦汉语排比句自动识别研究−以《孟子》《论语》中的排比句自动识别为例 (Study on automatic identification of parallel sentences in Pre-Qin Chinese − with automatic identification of parallel sentences in Mencius and the Analects of Confucius.).计算机工程与应用,2013,(19):222-226. DOI:10.3778/j.issn.1002-8331.1303-0471.

McHugh M. L. 2012. Interrater reliability: the kappa statistic. Biochemia medica, 22(3), 276–282. https://www.ncbi.nlm.nih.gov/pmc/articles/PMC3900052/?report=classic

穆婉青,廖健,王素格. 融合 CNN 和结构相似度计算的排比句识别及应用(A Combination of CNN and Structure Similarity for Parallelism Recognition)[J]. 中文信息学报, 2018, 32(2): 139-146. http://jcip.cipsc.org.cn/CN/abstract/abstract2525.shtml

Patrick Sturt, Frank Keller, Amit Dubey. 2010. Syntactic priming in comprehension: Parallelism effects with and without coordination, Journal of Memory and Language, Volume 62, Issue 4, , Pages 333-351, ISSN 0749-596X, https://doi.org/10.1016/j.jml.2010.01.001.

Saleema AbdulZahra Almehmdawi. 2018. Parallelism in One of Hillary Clinton's Speeches: A Critical Discourse Analysis. DOI: 10.18018/URUK/018-11/272-286.

Song, W., Liu, T., Fu, R., Liu, L., Wang, H., & Liu, T. 2016. Learning to Identify Sentence Parallelism in Student Essays. COLING.

Winfried Menninghaus, Valentin Wagner, Eugen Wassiliwizky, Thomas Jacobsen, Christine A. Knoop. 2017. The emotional and aesthetic powers of parallelistic diction, Poetics, Volume 63, Pages 47-59, ISSN 0304-422X, https://doi.org/10.1016/j.poetic.2016.12.001.

熊李艳,林晓乔,钟茂生.面向自动写作的中文排比句抽取方法(Automatic writing based on Chinese parallelism sentence extraction method). 计算机应用研究, 2018, 35(06):1751-1755.

Y. Dai, W. Song, X. Liu, L. Liu and X. Zhao, “Recognition of Parallelism Sentence Based on Recurrent Neural Network,” 2018. IEEE 9th International Conference on Software Engineering and Service Science (ICSESS), Beijing, China, pp. 148-151. doi: 10.1109/ICSESS.2018.8663734

张伯菁. (2002). Parallelism、antithesis 与排比、对偶. 湖北师范学院学报(哲学社会科学版), 22(4), 83-85. https://wenku.baidu.com/view/646ce845b307e87101f69648.html