Incremental Text Structuring with Word Clusters

Minh Quang Nhat Pham, Minh Le Nguyen, Akira Shimazu
School of Information Science
Japan Advanced Institute of Science and Technology
1-1 Asahidai, Nomi, Ishikawa 923-1292 Japan
{minhpqn,nguyenml,shimazu}@jaist.ac.jp

Abstract

The incremental text structuring is an incremental process in which one sentence is inserted into the existing document at each stage of the process, while preserving text coherence. In this paper, we present a method of incorporating additional features derived from word clusters for incremental text structuring tasks. We focus on the problem of lexical cohesion, using word cluster-based features to exploit semantic relations between words which are different in their surface representations but semantically related. Experimental results showed that combining word cluster-based features with baseline features will improve the performance of incremental text structuring.

1 Introduction

Text structuring is a subfield of natural language generation. Its purpose is to generate a coherent text based on a set of facts. Text structuring can be applied in multidocument summarization to reorder information derived from multi-text in appropriate order (Barzilay, 2003), or in community edited web resources on the internet like Wikipedia, or in news websites, etc. which require repeated updates when new information becomes available.

In collaboratively edited wikis, we often deal with the task of inserting new information into existing texts. This task takes much time and human efforts, especially in the cases of multiple texts or very long texts. Therefore, some tools that aid collaborative updating or automatically perform it could decrease maintenance efforts and improve document quality.

There are two approaches to the text structuring problem: sentence ordering (Lapata, 2003) which completely reorders a set of sentences into a text; and incremental text structuring (Chen et al., 2007) which inserts a sentence into an existing document at each stage. However, the problem setting in (Chen et al., 2007) may make readers confused about the reality of the insertion task. When editing a document, we normally read the document carefully before we add new sentences into it. Therefore, we often produce a sentence to add after we determine the location where the sentence is inserted. The reality of the insertion task will become explicable if we consider that it is to insert new information into an existing document; and a sentence is just one of the ways to represent information.

Chen et al. (2007) previously modeled the incremental text structuring problem as a hierarchical structure ranking problem, in which, to determine the best section (paragraph) for a sentence to be inserted, all sections (paragraphs) will be ranked by defined score and then, the section (paragraph) with the highest score will be chosen. Since the inserted sentence must be close to the topic of the surrounding sentences, the topical overlap between the inserted sentence and the candidate sections (paragraphs) is an important feature. In Chen et al. (2007), the TF-IDF weighting model was used to measure the topical overlap between an inserted sentence and surrounding sentences. However, only surface representations of words were used when computing topical overlap scores. This weighting model, therefore, cannot exploit relations between words which are different in surface representations but semantically related. Hence, it is attractive to consider the use of the intermediate representations of words rather than the words themselves to exploit the semantic similarity between words.

The idea of using intermediate representations of words has been explored by Miller et al. (2004) in Named-Entity Recognition tasks. That research

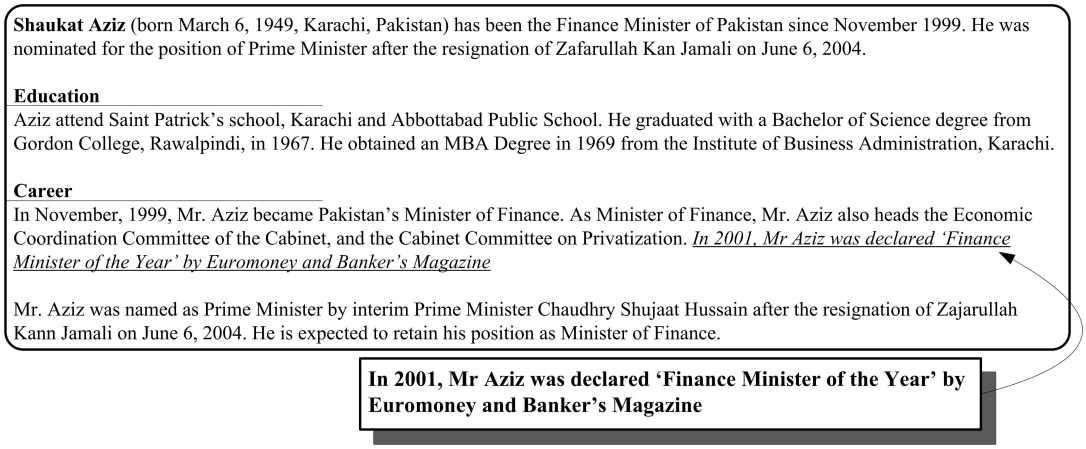

Figure 1: An example of Wikipedia insertion (Chen et al., 2007)

used additional word cluster-based lexical features, in which word clusters were obtained from a large unannotated corpus. Liang (2005) also used word cluster-based features in Chinese Word Segmentation and Named-Entity Recognition tasks. They showed that word clustering features improve performance of Named-Entity Recognition and Chinese Word Segmentation tasks. In dependency parsing, an important topic in natural language processing, Koo et al. (2008) demonstrated the effectiveness of using additional features that incorporate word clusters. The accuracy of dependency parsing with cluster-based features in the cases of English and Czech improved over the baseline accuracy.

In this paper, we extend the method using word cluster-based features to a new domain, the incremental text structuring task. Experiments showed the effectiveness of our approach against the baseline.

The remainder of this paper is divided as follows. Section 2 gives background on incremental text structuring and word clustering. Section 3 describes the baseline features and cluster-based features, Section 4 presents our experimental results, and Section 5 gives conclusions.

2 Background

2.1 Incremental Text Structuring

Text structuring is a subtask of natural language generation. It is a process of ordering a set of facts into a coherent text. The traditional approach to text structuring is sentence ordering (Lapata, 2003), which requires a complete reordering of the text in the question. Since sentence ordering approaches treat all sentences in the text equally, as new information becomes available, these method cannot take advantages of the existing text structure for the updating task.

Chen et al. (2007) introduced a new approach to text structuring. Text structuring was modeled as an incremental process. Instead of ordering sentences all at once, the text structuring process is divided into stages, and one new sentence is inserted into the existing document at each stage, while preserving document coherence. Specifically, the task of the sentence insertion system at each stage is to determine the best location in a text for a given input sentence. The insertion task was formulated as a hierarchical structured ranking problem to make use of the hierarchical structure of the existing text and to account for global coherence. The method was evaluated on real-world data obtained from Wikipedia articles. Figure 1 shows an example of Wikipedia insertion.

More formally, in the incremental text structuring problem, we are given a sequence of training instances. Each training instance is represented by three pieces of information (s, T, ℓ) in which s is a sentence, T is a document with hierarchical structure, and a leaf node ℓ represents the correct insertion point of the sentence s into T. Each sentence-node pair (s, n) where n is a node in T has an associated feature vector $\phi(s, n)$. The insertion point for an input sentence s is a leaf node ℓ chosen by taking into account its feature vector, and feature vectors of all its ancestors in the tree.

The model consists of a weight vector $\mathbf{w}$, each weight corresponds to a single feature. The leaf node ℓ, which is the insertion point for an input

sentence s, is determined by the following formula.

$$\arg\max_{\ell \in L(T)} \mathbf{w}.\Phi(s,\ell) \quad (1)$$

In Eq. 1, $L(T)$ is the set of leaf nodes in T and $\Phi(s,\ell)$ is the aggregate feature vector of a leaf node ℓ computed by the following Equation.

$$\Phi(s,\ell) = \sum_{n \in P(\ell)} \phi(s,n) \quad (2)$$

where $P(\ell)$ denotes the path from the root of the tree to a node ℓ.

Training procedure is implemented in an online learning framework. Like Perceptron update (Freund and Schapire, 1999), at each round, the model receives a training instance and predicts a leaf node according to the current parameters. Weights will be updated when the predicted leaf node is different from the correct leaf node. In the training algorithm, only weights found at the split point between predicted path and the true path are updated. The update rule for each round is defined as below.

$$\mathbf{w} \leftarrow \mathbf{w} + \phi\left(s, P(\ell)^{i^*+1}\right) - \phi\left(s, P(\hat{\ell})^{i^*+1}\right) \quad (3)$$

where $\hat{\ell}$ denotes the predicted leaf node, and ℓ is the correct leaf node; $P(\ell)^i$ denotes the i^{th} node on the path from the root to ℓ, and i^* is defined as the depth of the lowest common ancestor of ℓ and $\hat{\ell}$.

The key requirement of any text generation system is the coherence and the cohesion of its output. In the sentence insertion task, to preserve text cohesion, an inserted sentence has to be topically close to its surrounding sentences. Chen et al. (2007) previously measured topical overlap at the section level using TF-IDF weighted cosine similarity between an inserted sentence and a section. At the paragraph level, the topical overlap is computed in a similar way. However, when computing the topical overlap using TF-IDF weighting model, only surface representations of words were used. This weighting model therefore cannot exploit the relations between words which are different in surface representation but semantically related. From this observation, we decided to use word clusters as intermediate representations of words in our research to exploit these kinds of relations.

2.2 Brown Word Clustering

Word clustering is a process of assigning words to classes. Each class contains words which are semantically or syntactically similar. For example, the word *Thursday* is very much like the word *Friday* due to their function in expressing a day in a week. In natural language processing, word clustering can be used to tackle the problem of data sparseness by providing a lower-dimensional representation of words. The question is how to determine classes for words automatically. Brown et al. (1992) introduced statistical algorithms for assigning words to classes based on the frequency of their co-occurrence with other words in a large text data. Following is a brief description of the algorithm.

1001111011011	economic-consulting
1001111011011	investment-advisory
1001111011011	management-consulting
1001111011011	financial-planning
1001111011011	investment-management
...	
100111101011	electronics-parts
100111101011	vending-machine
100111101011	computer-peripherals
100111101011	industrial-electronics
...	
10100100010	legislator
10100100010	policeman
10100100010	soldier
10100100010	composer

Table 1: Examples of word clusters. Words having the same bit string representation belong to the same cluster

Brown word clustering algorithm received a vocabulary V of words to be assigned to classes and a text corpus as input. In the initial step, each word in the vocabulary V is assigned to a distinct class, and average mutual information between adjacent classes is computed. The algorithm then repeatedly merges the pairs of classes for which the loss in average mutual information is least. If C classes are required, $V - C$ merges need to be perfomed. The output of the algorithm is a binary tree, where each leaf node has a word and, each word occupies in only one leaf node. Words in the vocabulary can be represented more compactly by bit strings.

To conduct experiments in this paper, we used the word clusters of Koo et al. (2008) including 1000 word clusters. The Liang (2005) implementation of the Brown algorithm was used to obtain those word clusters. Table 1 provides some example bit strings.

Paragraph level features
TF score between *p* and *sen* based on nouns/proper nouns/verbs
TF-IDF score between *p* and *sen* based on nouns/proper nouns/verbs
Section level features
TF score between *sec* and *sen* based on nouns/proper nouns/verbs
TF-IDF score between *sec* and *sen* based on nouns/proper nouns/verbs

Table 2: The list of baseline topical overlap features (given an insertion sentence *sen*, and a paragraph *p* from a section *sec* of a document *d*) (Chen, 2008)

Paragraph level cluster-based features
TF score between *p* and *sen* based on bit string representations of nouns/proper nouns/verbs
TF-IDF score between *p* and *sen* based on bit string representations of nouns/proper nouns/verbs
Section level cluster-based features
TF score between *sec* and *sen* based on bit representation of nouns/proper nouns/verbs
TF-IDF score between *sec* and *sen* based on bit string representations of nouns/proper nouns/verbs

Table 3: List of features based on word clusters (given an insertion sentence *sen*, and a paragraph *p* from a section *sec* of a document *d*)

3 Features

In Chen et al. (2007), there are three types of features including: lexical features, positional features, and temporal features. Our research focuses on improving the accuracy of determining the insertion location in a text for a given piece of new information with word clusters as additional semantic resource. Therefore, in the following sections, first, we describe the topical overlap features which measure the topical closeness between an inserted sentence and surrounding sentences. Then, we present the additional cluster-based features.

3.1 Topical overlap features

As mentioned in Section 2.1, the topical overlap between an inserted sentence and surrounding sentences is one of the important features to preserve text cohesion of the original text. The topical overlap features at paragraph level and at section level previously were computed using the TF-IDF term weighting model. Table 2 provides the baseline topical overlap features (Chen, 2008).

3.2 Features based on word clusters

Since the baseline topical overlap features were computed using only surface representations of words in an input sentence and a text, we could not make use of the semantic relations between related words. In order to exploit these relations, we introduce a method to obtain additional features based on word clusters as follows.

Assume that we are given a set of word clusters W containing words along with their bit string representations as discussed shortly. For each pair of an insertion sentence *sen* and a paragraph *p*, we performed the following two steps:

Step 1: Obtain the bit string representation for each word in the insertion sentence *sen* and in the paragraph *p* from W. We omit words which are not included in any word clusters of the set W.

Step 2: Compute TF score and TF-IDF score between *sen* and *p* based on bit string representations of nouns/proper nouns/verbs.

At the section level, we computed the TF score and TF-IDF score based on word clusters for each pair of an insertion sentence and a section in the similar way.

Our approach relates to the concept of lexical cohesion (Jurafsky and Martin, 2008). Lexical cohesion is cohesion indicated by relations between words in the two units, such as use of identical words, a synonym, or hyponym. We illustrate that concept by an example from (Jurafsky and Martin, 2008).

(**Example 1**) *Peel, core and slice* ***the pears*** *and* ***the apples****. Add* ***the fruit*** *to the skillet.*

In this example, lexical cohesion between two sentences is indicated by the relation between the word *fruit* and the words *pears* and *apples*. In our research, we expect such kinds of relations can be captured using word clusters. Cluster-based topical overlap features are summarized in Table 3.

4 Experiments and Results

4.1 Experiment setting

Corpus and programs: In order to compare our method with the previous method, we used data

		Section	Paragraph	Insertion point	Tree Dist
Baseline	HIERARCHICAL	59.8	38.3	34.5	2.04
	PIPELINE	59.3	31.3	28.8	2.19
	FLAT	57.9	31.4	29.2	2.21
New-1	HIERARCHICAL	60.3 (+0.5)	39.2 (+0.9)	35.4 (+0.9)	2.01
	PIPELINE	60.4 (+1.1)	31.6 (+0.3)	29.1 (+0.3)	2.16
	FLAT	59.4 (+1.5)	35.5 (+4.1)	33.0 (+3.8)	2.10
New-2	HIERARCHICAL	*59.5*	39.3 (+1.0)	35.0 (+0.5)	2.02
	PIPELINE	59.3	31.7 (+0.5)	29.1 (+0.3)	2.18
	FLAT	59.3 (+1.4)	36.2 (+4.8)	33.7 (+4.5)	2.09
New-3	HIERARCHICAL	**60.2 (+0.4)**	**40.4 (+2.1)**	**36.1 (+1.6)**	**1.99**
	PIPELINE	60.4 (+1.1)	31.9 (+0.6)	29.3 (+0.5)	2.15
	FLAT	59.5 (+1.6)	36.2 (+4.8)	33.5 (+4.3)	2.08

Table 4: Insertion accuracy and tree distance of three methods in baseline setting and new settings. The last column gives tree distance. Shorter tree distance corresponds to better performance. New-1 = Only add cluster-based at section level; New-2 = Only add cluster-based features at paragraph level; New-3 = Add cluster-based features at both section level and paragraph level. Improvements of using cluster-based features in addition to baseline features are shown in parentheses.

and program in Chen et al. (2007)[1]. This data contains 4051 insertion/article pairs obtained from Wikipedia articles and update logs in the category "Living People." We also used 3240 pairs for training and 811 pairs for testing.

English word clusters: We used word clusters in Koo et al. (2008). The word clusters were derived from BLLIP corpus including 43 million words of Wall Street Journal text. There are 1000 clusters and 316710 word types in total.

Evaluation Measures: Two evaluation measures were used: a) insertion accuracy and b) the tree distance between the predicted and the true location of the inserted sentence. Insertion accuracy is the percentage of matches between predicted location of insertion and the true placement; and tree distance is defined as the length of the path through the tree which connects the predicted and the true paragraph positions. Shorter tree distance corresponds to better performance.

Baseline setting: In order to evaluate the effectiveness of using word cluster-based features as additional features, we compare our setting using cluster-based features with the baseline setting, which does not use cluster-based features. To do that, we computed accuracies of choosing sections, paragraphs, and *exact* insertion points, and then tree distances for the three methods presented in Chen et al. (2007): HIERARCHICAL, PIPELINE, and FLAT method. We will briefly describe three methods as follows.

In the HIERARCHICAL method, each document is represented as a hierarchy of sections, paragraphs, and sentences, and sentence insertion is operated over that hierarchical tree. Features are computed for each layer of the hierarchy. The new characteristic of the HIERACHICAL method is its update mechanism in the online learning framework to exploit hierarchical decomposition of features. Like hierarchical classification, the weight vector is learned for each node in the tree, and insertion decisions are based on all the weights along the path from node to root.

The PIPELINE method separately trains two rankers, one for section selection and one for paragraph selection. Like the pipeline mechanism, during decoding, the PIPELINE method first chooses the best section using the section-layer ranker, and then determines the best paragraph within the chosen section according to the paragraph-layer ranker.

The FLAT method makes use of all the same features as the HIERARCHICAL method, but it is trained with the standard ranking perceptron update (Collins, 2002), without making use of the hierarchical decompositions of features in Eq. 2.

New settings: To investigate how cluster-based features affect the insertion accuracy and the effectiveness of the method of formulating the sentence insertion task as a hierarchical structure ranking

[1]Code and data are freely available for download at http://people.csail.mit.edu/edc/emnlp07/

problem, we design three new settings in our experiments. The first setting adds cluster-based features only at the section level; the second setting adds cluster-based features only at the paragraph level; and the third setting adds cluster-based features at both the section level and the paragraph level.

4.2 Results and Discussion

Table 4 shows the performance of the baseline setting and three new settings. Except for the accuracy of choosing sections in the HIERARCHICAL method in the second setting, the three new settings outperform the baseline setting in terms of insertion accuracy and tree distance.

The results in Table 4 also indicate how features at section and paragraph levels affect insertion accuracy. In the first setting New-1, although we add cluster-based features only at the section level, the insertion accuracies at both the section level and the paragraph level increase. One can explain the reason for that by remembering that aggregate feature vectors at paragraph level are computed by summing all the feature vectors of nodes along the path from the current node to the root of the tree.

In the second setting New-2, when we add cluster-based features only at the paragraph level, the accuracy of choosing sections increases only in the FLAT method. The accuracy of choosing sections remains the same as for the baseline setting in the PIPELINE method, and slightly decreases in HIERARCHICAL method. However, insertion accuracies of choosing paragraphs and insertion points of the second setting New-2 are higher than for the baseline setting.

Among three new settings, in general, we obtain the best performance in the third setting New-3 which takes advantage of cluster-based features at both section level and paragraph level; and the HIERARCHICAL method in the third setting obtains the best performance in terms of choosing *exact* paragraphs and *exact* insertion points.

5 Conclusions

We have introduced a method of incorporating additional cluster-based features derived from a large unannotated text corpus for the incremental structuring task. Text coherence and cohesion of the output is the key criterion to evaluate the quality of text generation systems. In the incremental text structuring task, topical closeness between insertion sentences and surrounding sentences is often considered to preserve text cohesion. Experimental results showed the effectiveness and appropriateness of our approach to the sentence insertion task, and confirmed the advantage of using word clusters in terms of capturing topical overlap.

References

Regina Barzilay. 2003. *Information Fusion for Multidocument Summarization: Paraphrasing and Generation*. Ph.D. thesis, Columbia University.

Peter F. Brown, Vincent J. Della Pietra, Peter V. deSouza, Jenifer C. Lai, and Robert L. Mercer. 1992. Class-based n-gram models of natural language. *Computational Linguistics*, 18(4):467–479.

Erdong Chen. 2008. *Discourse Models for Collaboratively Edited Corpora*. Master's thesis, Massachusetts Institute of Technology.

Erdong Chen, Benjamin Snyder, and Regina Barzilay. 2007. Incremental text structuring with online hierachical ranking. In *Proceedings of the EMNLP*, pages 83–91.

Michael Collins. 2002. Discriminative training methods for hidden markov models: Theory and experiments with perceptron algorithms. In *Proceedings of the EMNLP*, pages 1–8.

Yoav Freund and Robert E. Schapire. 1999. Large margin classification using the perceptron algorithm. machine learning. *Machine Learning*, 37(3):277–296.

Daniel Jurafsky and James H. Martin. 2008. *Speech and Language Processing*. Prentice-Hall, New Jersey, USA.

Terry Koo, Xavier Carreras, and Michael Collins. 2008. Simple semi-supervised dependency parsing. In *Proceedings of ACL-08*, pages 595–603.

Mirella Lapata. 2003. Probabilistic text structuring: Experiments with sentence ordering. In *Proceedings of the ACL*, pages 545–552.

Percy Liang. 2005. *Semi-Supervised Learning for Natural Language*. Master's thesis, Massachusetts Institute of Technology.

Scott Miller, Jethran Guinness, and Alex Zamanian. 2004. Name tagging with word clusters and discriminative training. In *Proceedings of HLT-NAACL*, pages 337–342.