The Information Insertion Task with Intermediate Word Representation

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

1 Introduction

Nowadays, we are living in the society in which information is endlessly updated. For example, editors of newspapers or shared community on the Internet such as Wikipedia, always have to revise articles or write new ones when new information becomes available; personal websites are modified as status of the individual changes;

In legal domain, updating information is an important task because legal documents often change very fast and they must be updated consistently. One revision in a document may lead change requirements in related documents.

Our research copes with a special case of the updating task, the information insertion task which addresses the task of adding a piece of information into an existing document while preserving the continuity and coherence of the original text. In other words, the goal of this task is to determine the insertion location for new information into a document in which new information is assumed to be represented in a sentence.

Chen et al. (2007) modeled the information insertion task as a hierarchical structure ranking problem in the supervised learning framework. To determine the best paragraph for a sentence to be inserted, all paragraphs would be ranked by a ranking function which was learned from training data and then, the paragraph with the highest score will be chosen. Since the inserted sentence must be topically close to the surrounding sentences, the topical overlap between the inserted sentence and the candidate paragraphs is an important feature. The TF-IDF weighted cosine similarity was previously used to measure the topical overlap between two text segments. This weighting model only used surface forms of words, so the relations between words which are semantically related may not be exploited. Hence, it is attractive to consider the use of the intermediate representations of words rather than the words themselves when measuring semantic text similarity.

In our research, we investigate the processing models for the information insertion task on two datasets: Wikipedia insertion data obtained from Chen et al. (2007), and the Legal dataset built by ourselves. Next, we proposed a method of computing semantic text similarity based on intermediate word representations that incorporate word clusters, and apply to the information insertion task. Experiment results showed the effect of our method with improvements on both of datasets.

2 Background

2.1 Hierarchical ranking models

In the problem setting, we are given a set of training instances. Each training instance is represented by three pieces of information (s,T,ℓ) where s represents an input sentence, T is a hierarchically structured document, and ℓ represents the correct insertion location of the input sentence s in the document T. Each sentence-node pair (s,n) where n denotes a node in T, is associated with a feature vector $\phi(s,n)$. The insertion location for an input sentence s is a leaf node ℓ in the document tree, chosen by taking into account its associated 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.

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

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

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

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

Training procedure was 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, a heuristic update rule was proposed that 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 $\hat{\ell}$.

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. In natural language processing, word clustering can be used to tackle the problem of data sparseness by providing a lower-dimensional representation of words (Miller et al., 2004; Liang, 2005; Koo et al., 2008). 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.

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 binary 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.

3 Extracting Features based on Word Clustering

3.1 Basic idea

The simple lexical matching approach to computing topical overlap features was based on the *bag of words* assumption, using the surface forms of words in the input sentence and each candidate location. However, words in the input sentence may not appear in the locations, so the semantic relations between semantically related words are not exploited. In order to exploit these relations, we propose a method of using word clusters as intermediate word representations to obtain additional semantic features as follow.

Assume that we are given a set of word clusters W containing words in a vocabulary V along with their binary string representations as discussed above shortly. For each pair of an insertion sentence *sen* and a node *n* at a certain level in the document tree (*n* can be a paragraph or section in the case of Wikipedia data), we performed the following two steps:

Step 1: Obtain the binary string representation for each word in the insertion sentence *sen* and in the node *n* from W. Words which are not included in the vocabulary V will be assigned to a special value **null**.

Step 2: Compute the text similarity of two text segments *sen* and *n* based on their binary string representations.

When computing similarity scores of two text segments and using these scores as features in the

Paragraph level cluster-based features
TF score between p and *sen* based on binary string representations of non-stop-words/nouns/proper nouns/verbs
TF-IDF score between p and *sen* based on binary string representations of non-stop-words/nouns/proper nouns/verbs

Table 2: List of some sample features based on word clusters (given an insertion sentence *sen*, and a paragraph p)

learning model, we adopt three types of text similarity functions as follows.

3.2 Text similarity functions

TF-IDF weighted cosine similarity: We used TF-IDF weighted cosine similarity based on binary string forms of words. In effect, it is somewhat similar to the method of indexing using WordNet synsets (Gonzalo et al., 1998).

Lexical matching function: Lexical matching function measures the lexical-based semantic overlap of two text segments. To incorporate word clusters, we use the following formula to compute similarity of two text segments *sen* and n.

$$llm(sen,n) = \frac{\sum_{w \in sen} \omega(w,n)}{|sen|} \quad (4)$$

Where $|sen|$ denotes the number of words in the *sen*. The function $\omega(w,n)$ is an indicator function, returning the value 1 if w appears in n or there exists at least a word in n belonging to the same class with w.

Jaccard similarity function: The Jaccard similarity function of a sentence pair (s_1, s_2) is computed as below.

$$jsim(s_1,s_2) = \frac{\sum_{i \in s_1} \omega(i,s_2) + \sum_{j \in s_2} \omega(j,s_1)}{|s_1| + |s_2|} \quad (5)$$

Then, the text similarity of a sentence *sen* with a node n is computed by averaging out the similarity of all sentence in n with *sen*.

4 Experiment Setting

4.1 Data

Wikipedia insertion data: We obtained Wikipedia insertion data from 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 used 3240 pairs for training and 811 pairs for testing.

Legal data: Since there is no dataset for the information insertion task in Legal domain, we built the Legal dataset from the United States Code data[2]. All documents in the US Code data have highly hierarchically structures with sections as their basic coherent units. The dataset can be built by manually analyzing amendement parts of legal documents, but for evaluation purposes, we built the dataset automatically by recording documents before and after randomly removing a sentence from each of them. Totally, we obtained 1812 insertion sentence/document pairs from 18 legal documents. We used 1450 pairs (80%) for training and 362 (20%) 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.

4.2 Features

On Wikipedia insertion data, we used the feature set which was introduced in (Chen et al., 2007) as baseline features along with the features based on word clustering. The baseline features include of three types of features. The lexical features capture the topical overlap of an inserted sentence and a paragraph in a document; positional features aim to capture user preferences when adding new information into the body of a document; and the temporal features aim to make use of the chronological order of information in a Wikipedia document.

Since legal data have been built in a synthetic way, the positional features were not used. After analyzing data, we see that the temporal information is very rare in our dataset. Therefore, on legal data, only lexical features and word cluster-based features were extracted.

4.3 Evaluation measures

On Wikipedia data, we used the two evaluation measures: a) insertion accuracy and b) the tree distance between the predicted and the true location of the inserted sentence, in which tree distance is defined as the length of the path through the tree

[1]Data is freely available for download at http://people.csail.mit.edu/edc/emnlp07/

[2]The raw text of US Code is available on http://uscode.house.gov/lawrevisioncounsel.shtml

		Paragraph (%)	Tree Dist
Baseline	Flat	31.4	2.21
	Hier	38.3	2.04
New setting	Flat	36.2 (+4.8)	2.08
	Hier	40.4 (+2.1)	1.99

Table 3: The results on Wikipedia data

		Section (%)	*5*-best
Baseline	Flat	47.8	76.2
	Hier	50.9	81.8
New setting	Flat	49.5 (+1.7)	80.0 (+3.8)
	Hier	52.3 (+1.4)	83.0 (+1.2)

Table 4: The results on Legal data

which connects the predicted and the true paragraph positions. On the Legal data, we also used the insertion accuracy, and we introduced a new measure, *N*-best accuracy in which a prediction will be judged correct if the correct section is in the top *N* sections returned by the ranker.

4.4 Methods to evaluate

In order to evaluate the effect of using word cluster-based features, we compare our setting using cluster-based features with the baseline setting which does not incorporate cluster-based features. We evaluate the performance of two methods: the Hierarchical method as described in the Background section, and the Flat method using standard ranking perceptron update (Freund and Schapire, 1999), without making use of the hierarchical decompositions of features in Eq. 2.

5 Experiment Results

Table 3 shows the experiment results on the Wikipedia data, and the table 4 shows the results on the new Legal data. The results indicate that the Hierarchical method outperformed the Flat method on both the Wikipedia data and the new Legal data, and our method of combining cluster-based features with baseline features outperformed the baseline setting.

One problem of our method is that when the vocabulary of extracted word clusters is not large enough to cover words in the insertion datasets, some words may be omitted and relations of those words with others cannot exploited. Therefore, to be useful, the raw text data from which word clusters are extracted is required to be large enough to cover reasonably our insertion data.

6 Conclusions

We have introduced a method of incorporating cluster-based features derived from a large unlabeled text corpus. Despite differences between two datasets, experiment results showed improvements on both Wikipedia insertion dataset and the new Legal dataset.

The information insertion task is just a special case of the general updating task. As a future plan, we will deal with other kinds of the updating operations: deleting and modification.

Acknowledgement. This research was partially supported by the Ministry of Education, Science, Sports and Culture, COE Research and Grant-in-Aid for Scientific Research (B).

References

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

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

Julio Gonzalo, Felisa Verdejo, Irina Chugur, and Juan Cigarran. 1998. Indexing with wordnet synsets can improve text retrieval. In *Proceedings of the COLING/ACL'98 Workshop on Usage of WordNet for NLP*.

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

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.