A Regularized Compression Method To Unsupervised Word Segmentation

Ruey-Cheng Chen, **Chiung-Min Tsai** and **Jieh Hsiang**
National Taiwan University
1 Roosevelt Rd. Sec. 4
Taipei 106, Taiwan
rueycheng@turing.csie.ntu.edu.tw
cmtsai@mail.lis.ntu.edu.tw
jhsiang@ntu.edu.tw

Abstract

Languages are constantly evolving through their users due to the need to communicate more efficiently. Under this hypothesis, we formulate unsupervised word segmentation as a regularized compression process. We reduce this process to an optimization problem, and propose a greedy inclusion solution. Preliminary test results on the Bernstein-Ratner corpus and Bakeoff-2005 show that the our method is comparable to the state-of-the-art in terms of effectiveness and efficiency.

1 Introduction

Unsupervised word segmentation has been a popular research subject due to its close connection to language acquisition. It has attracted researchers from different communities, including linguistics, cognitive science, and machine learning, to investigate how human beings develop and harness their languages, and, more importantly, how knowledge is acquired.

In this paper we propose a new formulation to the unsupervised word segmentation problem. Our idea is based on the observation that language evolves because of the need to reduce communication efforts. For instance, new terminologies, abbreviations, and slang that carry complex semantics which cannot be efficiently expressed in the original languages are invented so that concepts can be conveyed. Such an evolution, we hypothesize, is limited to the extent where the evolved vocabulary exhibits similar complexity as the original one, in light of reducing the extra cost to pick up the new language. This process is realized as an optimization problem called *regularized compression*, which gets this name from its analogy to text compression.

The rest of the paper is organized as follows. We briefly summarize related work on unsupervised word segmentation in Section 2. In Section 3, we introduce the proposed formulation. The iterative algorithm and other technical details for solving the optimization problem are covered in Section 4. In Section 5, we describe the evaluation procedure and discuss the experimental results. Finally, we present concluding remarks in Section 6.

2 Related Work

The past few years have seen many nonparametric Bayesian methods developed to model natural languages. Many such applications were applied to word segmentation and have collectively reshaped the entire research field. Two most notable examples are hierarchical Bayesian models and the minimum description length principle. Our method fits in the latter category since we use this principle to optimize model parameters.

Hierarchical Bayesian methods were first introduced to complement conventional probabilistic methods to facilitate context-aware word generation. Goldwater et al. (2006) used hierarchical Dirichlet processes (HDP) to induce contextual word models. Their approach was a significant improvement over conventional probabilistic methods, and has inspired further explorations into more advanced hierarchical modeling techniques. Such examples include the nested Pitman-Yor process (Mochihashi et al., 2009), a sophisticated installment for hierarchical modeling at both word and character levels, and adaptor grammars (Johnson and Goldwater, 2009), a framework that aligns HDP to probabilistic context-free grammars.

The minimum description length (MDL) principle, originally developed in the context of information theory, was adopted in Bayesian statistics as a principled model selection method (Rissanen, 1978). Its connection to lexical acquisition was first uncovered in behavioral studies, and early applications focused mostly on applying MDL to induce word segmentation that results in compact lexicons (Kit and Wilks, 1999; Yu, 2000; Argamon et al., 2004). More recent approaches (Zhikov et al., 2010; Hewlett and Cohen, 2011) used MDL in combination with existing algorithms, such as branching entropy (Tanaka-Ishii, 2005; Jin and Ishii, 2006) and bootstrap voting experts (Hewlett and Cohen, 2009), to determine the best segmentation parameters. On various benchmarks, MDL-powered algorithms have achieved state-of-the-art performance, sometimes even surpassing that of the most sophisticated hierarchical modeling methods.

3 Regularized Compression

3.1 Preliminaries

Consider that the unsegmented text consists of K utterances and totally of N characters. We denote the text as a sequence of characters $\mathbf{c} = \langle c_1, \ldots, c_N \rangle$, as if conceptually concatenating all the K utterances into one string. The positions of all the utterance boundaries in $\mathbf{c}$ are represented as a set $U = \{u_0 = 0, u_1, \ldots, u_K\}$. In other words, the k-th utterance ($k = 1, \ldots, K$) is stored as the subsequence $\langle c_{u_{k-1}+1}, \ldots, c_{u_k} \rangle$ in $\mathbf{c}$.

A segmented text is denoted as a sequence of words $\mathbf{w} = \langle w_1, w_2, \ldots, w_M \rangle$ for some $M < N$. It represents the same piece of text as $\mathbf{c}$ does. The word sequence $\mathbf{w}$ is said to *respect* the utterance boundaries U if any word in the sequence does not span over two utterances. Unique elements in a character or word sequence implicitly define an alphabet set (or lexicon). Hereafter, we denote such alphabet sets for $\mathbf{c}$ and $\mathbf{w}$ as $A_{\mathbf{c}}$ and $A_{\mathbf{w}}$, respectively.

3.2 Effects of Compression

Word segmentation results from compressing a sequence of characters. By compression, we mean to replace the occurrences for some k-characters subsequence $\langle c_1, c_2, \ldots, c_k \rangle$ in the text with those for a new string $w = c_1 c_2 \ldots c_k$ (word). This procedure can be generalized to include more subsequences to be replaced, each with a different length. The resulting sequence is a mixture of characters and words introduced during compression. For clarity, we use the term *token sequence* to refer to such a mixed sequence of characters or words.

Compression has a few effects to the token sequence: (i) it increases the total number of tokens, (ii) it expands the alphabet set to include newly produced tokens, (iii) it affects the entropy rate estimates. Note that, by seeing a token sequence as a series of outcomes drawn from some underlying stochastic process, we can estimate the entropy rate empirically.

Items (i) and (ii) are natural consequences of compression. The effort to describe the same piece of information gets reduced at the expense of expanding the vocabulary, and sometimes even changing the usage. A real-life example for this is that language users invent new terminologies for efficiently conveying complex information. Item (iii) describes something more subtle. Observe that, when some n occurrences of a k-character subsequence $\langle c_1, c_2, \ldots, c_k \rangle$ get compressed, each character c_i loses n occurrences, and totally nk occurrences move away from the subsequence; as a result, the newly created word w receives n occurrences. It is clear that compression has this side effect of redistributing probability masses among the observations (i.e., characters), thereby causing deviation to entropy rate estimates.

3.3 Formulation

The choice of subsequences to be compressed is essential in the aforementioned process. We hypothesize that a good choice has the following two properties: (i) higher frequency, and (ii) low deviation in entropy rate.

We motivate these two properties as follows. First, high frequency subsequences are favorable here since they are more likely to be character-level collocations; compressing these subsequences results in better compression rate. Second, deviation in entropy rate is reflected in vocabulary complexity, and we believe that it directly translates to efforts that language users pay to adapt to the new language. In this case, there seems no reason to believe that either increasing or decreasing vocabulary complexity is beneficial, since in two trivial "bad choices" that one can easily imagine, i.e., the text being fully segmented or unsegmented, the entropy rates reach both extremes.

Motivated by these observations, we expect that the best word segmentation (i) achieves some predefined compression rate, and (ii) minimizes deviation in entropy rate. This idea is realized as an optimization problem, called *regularized compression*. Conceptually, this problem is defined as:

$$\begin{array}{ll} \text{minimize} & \mathrm{DV}(\mathbf{c}, \mathbf{w}) \\ \text{subject to} & \mathbf{w} \text{ respects } U \\ & \left|\frac{|\mathbf{w}|}{|\mathbf{c}|} - \rho\right| \le \epsilon \end{array} \qquad (1)$$

where ρ denotes some expected compression ratio and ϵ denotes the tolerance. Note that $\mathrm{DV}(\mathbf{c}, \mathbf{w}) = |\tilde{H}(C) - \tilde{H}(W)|$ represents the deviation in entropy rate with respect to sequences $\mathbf{c}$ and $\mathbf{w}$. In this definition, $\tilde{H}(C)$ and $\tilde{H}(W)$ denote the empirical entropy rates for random variables $C \in A_{\mathbf{c}}$ and $W \in A_{\mathbf{w}}$, estimated on the corresponding sequences $\mathbf{c}$ and $\mathbf{w}$, respectively.

4 Iterative Algorithm

4.1 Ordered Ruleset

Acknowledging that exponentially many feasible word sequences need to be checked, we propose an alternative formulation in a restricted solution space. The idea is, instead of optimizing for segmentations, we search for *segmentation generators*, i.e., a set of functions that generate segmentations from the input. The generators we consider here is the *ordered rulesets*.

An ordered ruleset $R = \langle r_1, r_2, \ldots, r_k \rangle$ is a sequence of translation rules, each of which takes the following form:

$$w \to c_1 c_2 \ldots c_n,$$

where the right-hand side ($c_1 c_2 \ldots c_n$) denotes the n-token subsequence to be replaced, and the left-hand side (w) denotes the new token to be introduced. Applying a translation rule r to a token sequence has an effect of replacing all the occurrences for subsequence $c_1 c_2 \ldots c_n$ with those for token w.

Applying an ordered ruleset R to a token sequence is equivalent to iteratively applying the translation rules $r_1, r_2, \ldots, r_k$ in strict order. Specifically, consider that the initial token sequence is denoted as $\mathbf{c}^{(0)}$ and let the final result be denoted as $\mathbf{c}^{(k)}$. By iterative application, we mean to repeat the following step for $i = 1 \ldots k$:

Apply rule r_i to $\mathbf{c}^{(i-1)}$ and save the result as $\mathbf{c}^{(i)}$.

4.2 Alternative Formulation

This notion of ordered rulesets allows one to explore the search space efficiently using a greedy inclusion algorithm. The idea is to maintain a globally best ruleset B that covers the best translation rules we have discovered so far, and then iteratively expand B by discovering new best rule and adding it to ruleset. The procedure repeats several times until the compression rate reaches some predefined ratio ρ. In each iteration, the best translation rule is determined by solving a modified version of Equation (1), which is written as follows:

$$\begin{array}{ll} \text{(In iteration } i\text{)} & \\ \text{minimize} & \alpha \frac{|\mathbf{c}^{(i)}|}{|\mathbf{c}^{(i-1)}|} + \mathrm{DV}(\mathbf{c}^{(i-1)}, \mathbf{c}^{(i)}) \\ \text{subject to} & r \text{ is a rule} \\ & r(\mathbf{c}^{(i-1)}) = \mathbf{c}^{(i)} \\ & \mathbf{c}^{(i)} \text{ respects } U \end{array} \qquad (2)$$

Note that the alternative formulation is largely a greedy version of Equation (1) except a few minor changes. First, the compression rate constraint becomes the termination condition in the greedy inclusion algorithm. Second, we add an extra term $|\mathbf{c}^{(i)}|/|\mathbf{c}^{(i-1)}|$ to the objective to encourage early inclusion of frequent collocations. The trade-off parameter α is introduced in Equation (2) to scalarize both terms in the objective.

A brief sketch of the algorithm is given in the following paragraphs.

1. Let B be an empty ordered ruleset, and let $\mathbf{c}^{(0)}$ be the original sequence of tokens.

2. Repeat the following steps for each $i \in \mathcal{N}$, starting from $i = 1$, until the compression rate reaches some predefined threshold.
    (a) Find a rule r that maximizes Equation (2)
    (b) Apply the rule r to form a new sequence $\mathbf{c}^{(i)}$ from $\mathbf{c}^{(i-1)}$.
    (c) Add r to the end of B.
3. Output B and the final sequence.

4.3 Implementation

Additional care needs to be taken in implementing Steps 2a and 2b. The simplest way to collect n-gram counts for computing the objective in Equation (2) is to run multiple scans over the entire sequence. Our experience suggests that using an indexing structure that keeps track of token positions can be more efficient. This is especially important when updating the affected n-gram counts in each iteration. Since replacing one occurrence for any subsequence affects only its surrounding n-grams, the total number of such affected n-gram occurrences in one iteration is linear in the number of occurrences for the replaced subsequence. Using an indexing structure in this case has the advantage to reduce seek time. Note that, however, the overall running time remains in the same complexity class regardless of the deployment of an indexing structure. The time complexity for this algorithm is $O(TN)$, where T is the number of iterations and N is the length of the input sequence.

Although it is theoretically appealing to create an n-gram search algorithm, in this preliminary study we used a simple bigram-based implementation for efficiency. We considered only bigrams in creating translation rules, expecting that the discovered bigrams can grow into trigrams or higher-order n-grams in the subsequent iterations. To allow unmerged tokens (i.e., characters that was supposed to be in one n-gram but eventually left out due to bigram implementation) being merged into the discovered bigram, we also required that that one of the two participating tokens at the right-hand side of any translation rule has to be an unmerged token. This has a side effect to exclude generation of collocation-based words[1]. It can be an issue in certain standards; on the test corpora we used, this kind of problems is not obvious.

[1]Fictional examples include "homework" or "cellphone".

Another constraint that we added to the implementation is to limit the choice of bigrams to those has more frequency counts. Generally, the number of occurrence for any candidate bigram being considered in the search space has to be greater or equal to some predefined threshold. In practice, we found little difference in performance for specifying any integer between 3 and 7 as the threshold; in this paper, we stick to 3.

5 Evaluation

5.1 Setup

We conducted a series of experiments to investigate the effectiveness of the proposed segmentation method under different language settings and segmentation standards. In the first and the second experiments, we focus on drawing comparison between our method and state-of-the-art approaches. The third experiment focuses on the influence of data size to segmentation accuracy.

Segmentation performance is assessed using standard metrics, such as precision, recall, and F-measure. Generally, these measures are reported only at word level; in some cases where further analysis is called for, we report boundary-level and type-level measures as well. We used the evaluation script in the official HDP package to calculate these numbers.

The reference methods we considered in the comparative study include the following:

- Hierarchical Dirichlet process, denoted as HDP (Goldwater et al., 2009);
- Nested Pitman-Yor process, denoted as NPY (Mochihashi et al., 2009);
- Adaptor grammars, denoted as AG (Johnson and Goldwater, 2009);
- Branching entropy + MDL, denoted as Ent-MDL (Zhikov et al., 2010);
- Bootstrap voting experts + MDL, denoted as BVE-MDL (Hewlett and Cohen, 2011);
- Description length gain, denoted as DLG (Zhao and Kit, 2008).

The proposed method is denoted as RC; it is also denoted as RC-MDL in a few cases where MDL is used for parameter estimation.

5.2 Parameter Estimation

There are two free parameters α and ρ in our model. The parameter α specifies the degree to which we favors high-frequency collocations when solving Equation (2). Experimentation suggests that α can be sensitive when set too low[2]. Practically, we recommend optimizing α based on grid search on development data, or the MDL principle. The formula for calculating description length is not shown here; see Zhikov et al. (2010), Hewlett and Cohen (2011), and Rissanen (1978) for details.

The expected compression rate ρ determines when to stop the segmentor. It is related to the expected word length: When the compression rate $|\mathbf{c}|/|\mathbf{w}|$ reaches ρ and the segmentor is about to stop, $1/\rho$ is the average word length in the segmentation. In this sense, it seems ρ is somehow connected to the language of concern. We expect that optimal values learned on one data set may thus generalize on the other sets of the same language. Throughout the experiments, we estimated this value based on development data.

5.3 Evaluation on Bernstein-Ratner Corpus

We conducted the first experiment on the Bernstein-Ratner corpus (Bernstein-Ratner, 1987), a standard benchmark for English phonetic segmentation. We used the version derived by Michael Brent, which is made available in the CHILDES database (Brent and Cartwright, 1996; MacWhinney and Snow, 1990). The corpus comprises 9,790 utterances, which amount to 95,809 words in total. Its relatively small size allows experimentation with the most computational-intensive Bayesian models.

Parameter estimation for the proposed method has been a challenge due to the lack of appropriate development data. We first obtained a rough estimate for the compression rate ρ via human inspection into the first 10 lines of the corpus (these 10 lines were later excluded in evaluation) and used that estimate to set up the termination condition. Since the first 10 lines are too small to reveal any useful segmentation cues other than the word/token ration of interest, we considered this setting ("almost unsupervised") a reasonable compromise. In this experiment, ρ is set to 0.37; the trade-off parameter α is set to 8.3, optimized using MDL principle in a two-pass grid search (the first pass over $\{1, 2, \ldots, 20\}$ and the second over $\{8.0, 8.1, \ldots, 10.0\}$).

[2]Informally speaking, when $\alpha < \tilde{H}(\mathbf{c})$. The analysis is not covered in this preliminary study.

	P	R	F	Time
HDP	0.752	0.696	0.723	–
NPY, bigram	0.748	0.767	0.757	17 min.
AG	–	–	**0.890**	–
Ent-MDL	0.763	0.745	0.754	2.6 sec.
BVE-MDL	**0.793**	0.734	0.762	2.6 sec.
RC-MDL	0.771	**0.819**	0.794	0.9 sec.

Table 2: Performance evaluation on the Bernstein-Ratner corpus. The reported values for each method indicate word precision, recall, F-measure and running time, respectively. The boldface value for each column indicates the top performer under the corresponding metric.

A detailed performance result for the proposed method is described in Table 1. A reference run for HDP is included for comparison. The proposed method achieved satisfactory result at word and boundary levels. Nevertheless, low type-level numbers (in contrast to those for HDP) together with high boundary recall suggested that we might have experienced over-segmentation.

Table 2 covers the same result with less details in order to compare with other reference methods. All the reported measures for reference methods are directly taken from the literature. The result shows that AG achieved the best performance in F-measure (other metrics are not reported), surpassing all the other methods by a large margin (10 percent). Among the other methods, our method paired with MDL achieved comparable performance as the others in precision; it does slightly better than the others in recall (5 percent) and F-measure (2.5 percent). Furthermore, our algorithm also seems to be competitive in terms of computational efficiency. On this benchmark it demanded only minimal memory low as 4MB and finished the segmentation run in 0.9 second, even less than the reported running time for both MDL-based algorithms.

	P	R	F	BP	BR	BF	TP	TR	TF
HDP, Bernstein-Ratner	0.75	0.70	0.72	0.90	0.81	0.85	0.64	0.55	0.59
RC-MDL, Bernstein-Ratner	0.77	0.82	0.79	0.85	0.92	0.89	0.57	0.48	0.50
RC, CityU training	0.75	0.79	0.77	0.89	0.93	0.91	0.63	0.35	0.45
RC, MSR training	0.73	0.82	0.77	0.86	0.96	0.91	0.70	0.26	0.38

Table 1: Performance evaluation for the proposed method across different test corpora. The first row indicates a reference HDP run (Goldwater et al., 2009); the other rows represent the proposed method tested on different test corpora. Columns indicates performance metrics, which correspond to precision, recall, and F-measure at word (P/R/F), boundary (BP/BR/BF), and type (TP/TR/TF) levels.

Corpus	Training (W/T)	Test (W/T)
AS	5.45M / 141K	122K / 19K
PKU	1.1M / 55K	104K / 13K
CityU	1.46M / 69K	41K / 9K
MSR	2.37M / 88K	107K / 13K

Table 3: A short summary about the subsets in the Bakeoff-2005 dataset. The size of each subset is given in number of words (W) and number of unique word types (T).

	CityU	MSR
RC, $r = 0.65$	0.770	0.774
DLG, ensemble	0.684	0.665
Ent-MDL, $n_{max} = 3$	**0.798**	**0.795**

Table 4: Performance evaluation on the common training subsets in the Bakeoff-2005 and Bakeoff-2006 datasets. The reported values are token F-measure. The boldface value in each column indicates the top performer for the corresponding set.

5.4 Evaluation on Bakeoff-2005 Corpus

The second benchmark that we adopted is the SIGHAN Bakeoff-2005 dataset (Emerson, 2005) for Chinese word segmentation. The corpus has four separates subsets prepared by different research groups; it is among the largest word segmentation benchmarks available. Table 3 briefly summarizes the statistics regarding this dataset.

We decided to compare our algorithm with description length gain (DLG), for that it seems to deliver best segmentation accuracy among other unsupervised approaches ever reported on this benchmark (Zhao and Kit, 2008). Since the reported values for DLG were obtained on another closed dataset Bakeoff-2006 (Levow, 2006), we followed a similar experimental setup as suggested in the literature (Mochihashi et al., 2009): We compared both methods only on the training sets for the common subsets CityU and MSR. Note that this experimental setup departed slightly from that of Mochihashi et al. in that all the comparisons were strictly made on the training sets. The approach is more straightforward than the suggested sampling-based method.

Other baseline methods that we considered include HDP, Ent-MDL, and BVE-MDL, for their representativeness in segmentation performance and ease of implementation. The HDP implementation we used is a modified version of the offical HDP package[3]; we patched the package to make it work with Unicode-encoded Chinese characters. For Ent-MDL and BVE-MDL, we used the software package[4] distributed by Hewlett and Cohen (2011). We estimated the parameters using the AS training set as the development data. We set α to 6 based on a grid search. The expected compression rate ρ that we learned from the development data is 0.65.

In Table 1, we give a detailed listing of various performance measures for the proposed method. Segmentation performance seems moderate at both word and boundary levels. Nevertheless, high type precision and low type recall on both CityU and MSR training corpora signaled that our algorithm failed to discover most word types. This issue, we suspect, was caused by exclusion of low-frequency candidate bigrams, as discussed in Section 4.3.

Table 4 summarizes the result for word segmentation conducted on the CityU and MSR subsets of Bakeoff-2005. Due to practical computational limits, we were not able to run HDP and BVE-MDL on any complete subset. The result shows that our

[3]http://homepages.inf.ed.ac.uk/sgwater/

[4]http://code.google.com/p/voting-experts

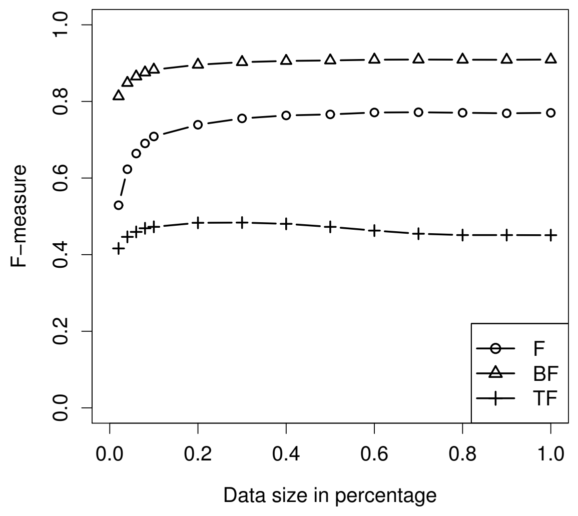

Figure 1: Performance evaluation for the proposed method on the CityU training set.

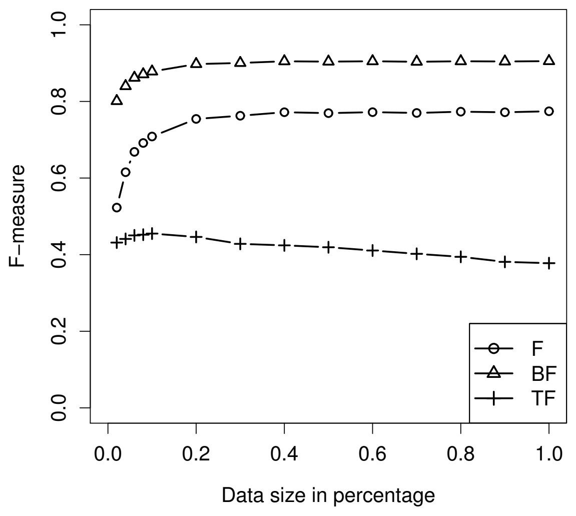

Figure 2: Performance evaluation for the proposed method on the MSR training set.

	CityU-1k	MSR-1k
RC, $r = 0.65$	0.505	0.492
HDP, 10 sample average	0.591	**0.623**
RC, $r = 0.65$/punc.	**0.599**	0.591

Table 5: Performance evaluation on two random samples from the common sets (CityU and MSR subsets) in the Bakeoff-2005 and Bakeoff-2006 datasets.

algorithm outperforms DLG by 8 to 10 percents in F-measure, while Ent-MDL still performs slightly better, achieving the top performance among all the experimental runs on both subsets.

To compare with HDP, we conducted another test run on top of a random sample of 1,000 lines from each subset. We chose 1,000 lines because HDP can easily consume more than 4GB of main memory on any larger sample. We adopted standard settings for HDP: $\alpha_0 = 3,000$, $\alpha_1 = 300$, and $p_b = 0.2$. In each trial run, we ran the Gibbs sampler for 20,000 iterations using simulated annealing (Goldwater et al., 2009). We obtained 10 samples from the Gibbs sampler and used the average performance in comparison. It took slightly more than 50 hours to collect one trial run on one subset.

The evaluation result is summarized in Table 5. We ran our algorithm to the desired compression ratio $r = 0.65$ on this small sample. The result shows that the performance of regularized compression is inferior to that of HDP by 9 to 13 percents in F-measure for both sets. To investigate why, we looked into the segmentation output. We observed that, in the regularized compression output, most of the punctuation marks were incorrectly aligned to their neighboring words, owing to the short of frequency counts in this small sample. The HDP, however, does not seem to suffer from this issue.

We devised a simple post-processing step, in which each punctuation mark was forced segmented from the surrounding text. Another outside test was conducted to see how well the algorithm works using heuristics derived from minimal domain knowledge. The additional run is denoted as RC/punc. The result is shown in Table 5. From the result, we found that the combined approach works slightly better than HDP in one corpus, but not in the other.

5.5 Effects of Data Size

We employed the third experiment to study the influence of corpora size to segmentation accuracy. Since the proposed method relies on empirical estimates for entropy rate to decide the word boundaries, we were interested in learning about how it responds to relatively low and high volume input.

This experiment was conducted on CityU and

MSR training sets. On each corpus, we took the first $k\%$ of data (in terms of utterances) and tested the proposed method against that subset; this test was repeated several times with different values for k. In this experiment, we chose the value for k from the set $\{2, 4, 6, 8, 10, 20, 30, \ldots, 90, 100\}$. The performance is evaluated using word, boundary, and type F-measures.

Figures 1 and 2 show the experiment results. Both figures revealed similar patterns for segmentation performance at different volume levels. Word F-measures for both corpora begin at roughly 0.52, climb up rapidly to 0.73 as the volume grows from 2% to 20%, and finally settle on some value around 0.77. Boundary F-measures for both corpora show a similar trend—a less steep increase before 20% from 0.80 to 0.89 followed by a plateau at around 0.93. Here, the result seems to suggest that estimating token entropy rate using less than 20% of data might be insufficient for this type of text corpora. Furthermore, since performance is saturated at such an early stage, it seems feasible to split the entire dataset into a number of folds (e.g., 5, in this case) and solve each fold individually in parallel. This technique may greatly enhance the run-time efficiency of the segmentor.

The patterns we observed for type F-measure tells another story. On both corpora, type F-measures do not seem to improve as data volume increases. On CityU corpora, type F-measure gradually increased from 0.42 to 0.48 and then slowly falling back to 0.45. On MSR corpora, type F-measure peaked at 0.45 when receiving 10% of data; after that it started decreasing, going all the way down to 0.37, even lower than the number 0.43 it received at the beginning. Our guess is that, at some early point (20%), the proposed method started to under-segment the text. We suspect that there is some deep connection between performance saturation and under-segmentation, since from the result they both begin at roughly the same level. Further investigation in this respect is needed to give out definitive explanations.

6 Concluding Remarks

Preliminary experimental results suggest that the regularized compression method, even only with partial evidence, seems as effective as the state-of-the-art methods in different language settings. When paired with MDL criteria, regularized compression is comparable to hierarchical Bayesian methods and MDL-based algorithms in terms of segmentation accuracy and computational efficiency. Furthermore, regularized compression is less memory-demanding than the other approaches; thus, it scales more easily to large corpora for carrying out certain tasks such as segmenting historical texts written in ancient languages, or preprocessing a large dataset for subsequent manual annotation.

We have identified a number of limitations of regular compression. First, the choice of candidate n-grams does not cover *hapax legomena*, i.e., words that occur only once in the corpus. At present, precluding these low-frequency n-grams seems to be a necessary compromise due to our limited understanding about the dynamics behind regular compression. Second, regularized compression does not work well with low volume data, since on smaller dataset the distribution of frequency counts is less precise. Third, the algorithm may stop identifying new word types at some point. We suspect that this is related to the choice of n-gram, since in our implementation no two existing "words" can be aggregated into one. These issues shall be addressed in our future work.

Acknowledgments

We thank the anonymous reviewers for their valuable comments. The research efforts described in this paper are supported under the National Taiwan University Digital Archives Project (Project No. NSC-98-2631-H-002-005), which is sponsored by National Science Council, Taiwan.

References

Shlomo Argamon, Navot Akiva, Amihood Amir, and Oren Kapah. 2004. Efficient unsupervised recursive word segmentation using minimum description length. In *Proceedings of the 20th international conference on Computational Linguistics*, COLING '04, Stroudsburg, PA, USA. Association for Computational Linguistics.

Nan Bernstein-Ratner. 1987. The phonology of parent child speech. *Children's language*, 6:159–174.

Michael R. Brent and Timothy A. Cartwright. 1996. Distributional regularity and phonotactic constraints are useful for segmentation. In *Cognition*, pages 93–125.

Thomas Emerson. 2005. The second international chinese word segmentation bakeoff. In *Proceedings of the Fourth SIGHAN Workshop on Chinese Language Processing*, volume 133. Jeju Island, Korea.

Sharon Goldwater, Thomas L. Griffiths, and Mark Johnson. 2006. Contextual dependencies in unsupervised word segmentation. In *Proceedings of the 21st International Conference on Computational Linguistics and the 44th annual meeting of the Association for Computational Linguistics*, ACL-44, pages 673–680, Stroudsburg, PA, USA. Association for Computational Linguistics.

Sharon Goldwater, Thomas L. Griffiths, and Mark Johnson. 2009. A bayesian framework for word segmentation: Exploring the effects of context. *Cognition*, 112(1):21–54, July.

Daniel Hewlett and Paul Cohen. 2009. Bootstrap voting experts. In *Proceedings of the 21st international jont conference on Artifical intelligence*, IJCAI'09, pages 1071–1076, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc.

Daniel Hewlett and Paul Cohen. 2011. Fully unsupervised word segmentation with BVE and MDL. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies: short papers - Volume 2*, HLT '11, pages 540–545, Stroudsburg, PA, USA. Association for Computational Linguistics.

Zhihui Jin and Kumiko T. Ishii. 2006. Unsupervised segmentation of chinese text by use of branching entropy. In *Proceedings of the COLING/ACL on Main conference poster sessions*, COLING-ACL '06, pages 428–435, Stroudsburg, PA, USA. Association for Computational Linguistics.

Mark Johnson and Sharon Goldwater. 2009. Improving nonparameteric bayesian inference: experiments on unsupervised word segmentation with adaptor grammars. In *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, NAACL '09, pages 317–325, Stroudsburg, PA, USA. Association for Computational Linguistics.

Chunyu Kit and Yorick Wilks. 1999. Unsupervised learning of word boundary with description length gain. In *CoNLL-99*, pages 1–6, Bergen, Norway.

Gina-Anne Levow. 2006. The third international chinese language processing bakeoff: Word segmentation and named entity recognition. In *Proceedings of the Fifth SIGHAN Workshop on Chinese Language Processing*, volume 117. Sydney: July.

Brian MacWhinney and Catherine Snow. 1990. The child language data exchange system: an update. *Journal of child language*, 17(2):457–472, June.

Daichi Mochihashi, Takeshi Yamada, and Naonori Ueda. 2009. Bayesian unsupervised word segmentation with nested Pitman-Yor language modeling. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP: Volume 1 - Volume 1*, ACL '09, pages 100–108, Stroudsburg, PA, USA. Association for Computational Linguistics.

Jorma Rissanen. 1978. Modeling by shortest data description. *Automatica*, 14(5):465–471, September.

Kumiko Tanaka-Ishii. 2005. Entropy as an indicator of context boundaries: An experiment using a web search engine. In Robert Dale, Kam-Fai Wong, Jian Su, and Oi Kwong, editors, *Natural Language Processing IJCNLP 2005*, volume 3651 of *Lecture Notes in Computer Science*, chapter 9, pages 93–105. Springer Berlin / Heidelberg, Berlin, Heidelberg.

Hua Yu. 2000. Unsupervised word induction using MDL criterion. In *Proceedings of the International Symposium of Chinese Spoken Language Processing*, Beijin, China.

Hai Zhao and Chunyu Kit. 2008. An empirical comparison of goodness measures for unsupervised chinese word segmentation with a unified framework. In *The Third International Joint Conference on Natural Language Processing (IJCNLP-2008)*.

Valentin Zhikov, Hiroya Takamura, and Manabu Okumura. 2010. An efficient algorithm for unsupervised word segmentation with branching entropy and MDL. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, EMNLP '10, pages 832–842, Stroudsburg, PA, USA. Association for Computational Linguistics.