Unsupervised Text Pattern Learning Using Minimum Description Length

Wu Ke[1], Yu Jiangsheng[1]

Department of Computer Science and Technology, Peking University

Abstract. The knowledge of text patterns in a domain-specific corpus is valuable in many natural language processing (NLP) applications such as information extraction and question-answering system. In this paper, we propose a simple but effective probabilistic language model for modeling the in-decomposability of text patterns. Under the minimum description length (MDL) principle, an efficient unsupervised learning algorithm is implemented and the experiment on an English critical writing corpus has shown promising coverage of patterns compared with human summary.

1 Introduction

A text pattern is a fixed way to formulate a whole phrase, clause or sentence with a strong in-decomposability of use. For example, the following are patterns extracted from an English critical writing corpus:

- In the argument [... *more about the argument* ...]
- To better assess the argument I would need [... *some details* ...]
- It is entirely possible [... *possibility 1* ...] or [... *possibility 2* ...]
- To strengthen it [... *appellation* ...] must provide/show [... *some details* ...]

Such text patterns are useful in many NLP applications such as information extraction [1] and question-answering system [2]. Many methods for learning patterns are proposed in, for example, [1] and [3]. However, most of those approaches assume a set of training data which contains only sentences from a single pattern or a group of several similar patterns. Therefore, it is desired that pattern learning could be done upon general and wide coverage data with little prior knowledge of the corpus.

This paper addresses this problem with a probabilistic language model and uses the minimum description length (MDL) principle proposed by [4] for unsupervised learning. Our goal is to learn as many as possible fixed parts of text patterns (for example, we wish to learn "to strength it ... must show ..." but do not expect to be able to tell that the first ellipsis should be replaced with words or phrases similar to "the author" while the second one should be replaced with some detailed descriptions). The fundamental intuition is that since patterns exhibit strong inside in-decomposability, their statistical features should be more

close to words than ordinary phrases. By merging tokens which we believe to be in-decomposable, a dictionary containing all merged patterns and unmerged words could be obtained. An effective hill-climbing search algorithm is designed for learning the dictionary. Experiment results show that our approach is able to learn a considerable proportion of typical patterns from a corpus within a general category.

2 The Probabilistic Language Model

For convenience of discussion, we first formalize some related concepts to text patterns.

Definition 1. *We consider the text as a sequence of basic symbols drawn from a discrete and finite set called the* alphabet*. Any member of an alphabet X is called a* letter *of X. A* word *of X is the concatenation of letters in X. The data is in the form of a concatenation of words and the word boundaries are hidden for us to discover.*

Definition 2. *For some alphabet X with a word set W, If a function $f : (W^*)^n \to W^*$ satisfies*

$$f(s_1, s_2, \ldots, s_n) = \pi_0 s_1 \pi_1 s_2 \pi_2 \ldots \pi_{n-1} s_n \pi_n \ ,$$

*where $\pi_0, \pi_1, \ldots, \pi_n$ are strings over X, then f is called an n-*ary text pattern *over X.*

Now we can restate our goal using above terms — to learn the $(\pi_0, \pi_1, \ldots, \pi_n)$ of all f's.

Suppose our corpus is part of some unknown language $\mathcal{L}$ on a known alphabet X. We assume that $\mathcal{L}$ has a finite vocabulary called a *dictionary*.

Definition 3. *The* dictionary *of $\mathcal{L}$ is a finite subset of X^+. All strings in $\mathcal{L}$ can and only can use words in D.*

Furthermore, we attach a probability mass function to a dictionary.

Definition 4. *For a given dictionary D, if function $\theta_D : X^+ \to \mathbb{R}$ satisfies*

1. $\forall x \in D,\ 0 \leq \theta_D(x) \leq 1$,
2. $\forall x \in X^+ - D,\ \theta_D(x) = 0$,
3. $\sum_{x \in D} \theta_D(x) = 1$,

then θ_D is called a dictionary probability source of D.

θ_D accounts for the probability of strings with only one word. Further more we assume that all draws of words are independent and identically distributed (i.i.d.) to θ_D. This finalizes our language model.

Definition 5. ***(Language Model)*** *Given a language $\mathcal{L} = (X, D, \theta_D)$, a string in $\mathcal{L}$ is generated by drawing words from D i.i.d. The sequence $x^n = x_1 x_2 \dots x_n \in X^+$ is obtained by concatenating its original word sequence $w^m = w_1 w_2 \dots w_m$. Supposing the existence of a one-to-one mapping[1] between x^n and w^m for all $x^n \in \mathcal{L}$, the probability of the observation $x^n = w^m$ is $P(x^n) = \prod_{i=1}^{m} \theta_D(w_i)$. If such mapping does not exist, then $x^n \notin \mathcal{L}$, therefore $P(x^n) = 0$.*

The maximum likelihood (ML) estimator for the dictionary probability source θ_D has a simple form.

Theorem 1. *For observation $x^n = w^m = w_1 w_2 \dots w_m$ and a dictionary D, the ML estimator for θ_D is*

$$\hat{\theta}_D(w_i) = \frac{c(w_i)}{m} \ , \ \forall w_i \in D \ , \tag{1}$$

where $c(w_i)$ denotes the number of appearances of word w_i in w^m.

2.1 An Interpretation of the Model

When the dictionary is known, our model is essentially equivalent to a uni-gram language model. One of the major differences of our model from an ordinary uni-gram model is the discrimination of sample spaces — our model sees a letter from the alphabet X as an elementary event while a uni-gram model sees a word as an elementary event. This discrimination is necessary since we have to stick to a static sample space while searching for different dictionaries. Therefore, a two-level letter-word treatment is necessary instead of the simple uni-gram one. At the same time, by seeing patterns as a whole "word", we push the in-decomposability within a pattern to the extreme — the constituents of a string are either i.i.d. words or strongly inside correlated patterns. As a result, merging in-decomposable tokens serves as the way to search for a better-fit dictionary to data.

3 The Minimum Description Length Principle

Naïvely maximizing the goodness-of-fit, or likelihood, of is a trivial but not effective approach to our dictionary selection task. To see this, consider a one-big-word dictionary $D' = \{x^n\}$. With Theorem 1 one can see that $P(x^n|D', \hat{\theta}_{D'}) = 1$. Moreover, $P(x^n|D'', \hat{\theta}_{D''}) < 1$ holds for every dictionary D'' that explains x^n with more than one single word. As a result, no matter what search strategy we use, an over-fitting and thus useless "optimal" is inevitable. Fortunately, we find that the minimum description length principle could to some degree tackle this problem.

[1] As mention in Section 1, we are using this model during a dictionary building process starting with each letter a dictionary item, in which case such a mapping is guaranteed to exist.

The minimum description length (MDL) principle is proposed in [4] as a general model selection framework. It has been used in many unsupervised learning tasks, for example, [5] used it for word segmentation and [6] used it for morphology learning. This section gives a brief introduction about MDL from a perspective closely related to our problem. For a more general cover on this topic, there are comprehensive materials such as [7]. In brief, MDL could be described as follows,

Definition 6. *Let $\mathcal{H}$ be the set of all candidate models. The best model $H \in \mathcal{H}$ to explain the data x^n is the one that minimizes the sum*

$$L(H) - \log P(x^n|H, \hat{\theta}_H) \ , \tag{2}$$

where $L(H)$ is the model length in bits and $P(x^n|H, \hat{\theta}_H)$ is the maximum likelihood of x^n under H, which is also known as the data length $L(x^n|H)$.

Supposing $\sum_{H\in\mathcal{H}} 2^{-L(H)} = c < \infty$, $\frac{2^{-L(H)}}{c}$ could be seen as a prior of H preferring shorter models, then minimizing (2) is equivalent to maximizing the posterior probability $P(H|X)$. This is the Bayesian interpretation of MDL, and interpretations from other perspectives could be found in [7] and [8].

MDL does not give a precise way of evaluating $L(H)$ and many methods, either specific or general, have been proposed. Some sophisticated method (such as the stochastic parametric complexity using the normalized maximum likelihood (NML) from [9]) are capable of model selection from different families of models. However, in many simpler tasks, where candidate models are from the same family, the model length function $L(H)$ could be defined in some much more straight-forward manner called crude MDL. In these cases, [8] showed that many straight-forwardly designed model length functions, conforming to several basic restrictions, are statistically consistent and convergent, which may serve as a guideline for designing model length functions.

4 Crude MDL for Learning Patterns

4.1 Length Functions

As mentioned in Section 2, the data is seen as a sequence of letters over some given alphabet X generated by some unknown dictionary D and corresponding dictionary probability source θ_D. Our major concern is to find the dictionary D. To start, we need a model length function. A straightforward code scheme for a dictionary is to encode a list of all dictionary items separated by an additional separator symbol. Thus we have $|X| + 1$ symbols to encode and a uniform code scheme will encode each symbol in $\log(|X| + 1)$ bits. Let $\lambda(D)$ be the sum of word lengths in D, then

$$L(D) = (\lambda(D) + |D|) \log(|X| + 1) \ . \tag{3}$$

For example, when $X = \{a, b\}, D = \{a, ab\}$, the dictionary D is represented as "$a; ab;$" where ";" is the additional separator. And the code length for $a; ab$ is $5 \log 3$.

With the help of Theorem 1, we could derive the data length of x^n given the dictionary D as follows,

$$\begin{aligned} L(x^n|D) &= -\log P(x^n|D,\hat{\theta}_D) \ , \\ &= -\sum_{i=1}^{\Omega} \log \hat{\theta}_D(w_i) \ , \\ &= -\sum_{w\in D} c(w) \log \frac{c(w)}{\Omega} \ , \\ &= \Omega \log \Omega - \sum_{w\in D} c(w) \log c(w) \ , \end{aligned} \tag{4}$$

where Ω denotes the number of words of x^n under dictionary D, and $c(w)$ is the number of appearances of word w in x^n.

4.2 Continuous and Non-continuous Patterns

Text patterns defined in Definition 2 can be further divided into two classes,

Continuous pattern A pattern in the form $\pi_0 s_1$, i.e. a pattern with only one fixed part, for example "in the argument ...";

Non-continuous pattern A pattern with at least two fixed parts, such as $\pi_0 s_1 \pi_1$, for example "it is entirely possible ... or ...".

The representation of a continuous pattern in a dictionary is trivially its only fixed part. However, things are not as trivial with a non-continuous pattern — storing each fixed part a dictionary item obviously do not reflect their indecomposability and traditional treatment of n-gram models with a constant n does not fit the model and has difficulty in handling long distance dependencies. Nevertheless, by exploiting the exchangeability between words offered by our language model, this problem can be solved. Let

$$f(s_1, s_2, \ldots, s_n) = \pi_0 s_1 \pi_1 s_2 \pi_2 \ldots \pi_{n-1} s_n \pi_n \ , \tag{5}$$

be a non-continuous pattern that we wish to learn, then we may carry out a transformation on the data, substituting every $\pi_0 s_1 \pi_1 s_2 \pi_2 \ldots \pi_{n-1} s_n \pi_n$ with $\pi_0 \pi_1 \ldots \pi_n s_1 s_2 \ldots s_n$. After this, this new pattern looks like

$$f'(s_1, s_2, \ldots, s_n) = \pi_0 \pi_1 \ldots \pi_n s_1 s_2 \ldots s_n \ , \tag{6}$$

which is actually now a continuous pattern

$$f''(s_1) = \pi_0 \pi_1 \ldots \pi_n s_1 \ . \tag{7}$$

The i.i.d. assumption of our language model ensures such exchangeability — the transformation does not change to probability of data. In fact, this is a important reason for us to choose such a language model.

4.3 Algorithm Design

To start with, the dictionary that contains exactly all the letters used in the data is chosen, since this is the most general dictionary that is able to explain the data. Then from this initial dictionary, we search towards a more specific dictionary via merging an ordered pair of words in the old dictionary. Within the range of a sentence, we apply the transformation in Section 4.2 and merge all appearances of the word pair in data. The pair which maximizes the compression of description length $L(D) + L(x^n|D) - L(D') - L(x^n|D')$ is chosen, where D' is the new dictionary got by merging. Such process is repeated until we cannot further compress the description length of x^n.

The fundamental steps of the algorithm are listed in Figure 1.

1. **Initialization** Start with $D \leftarrow X$ where X is the minimum alphabet to cover x^n. The list $w \leftarrow [x_1, x_2, \ldots, x_n]$ is the segmentation of x^n under dictionary D.
2. **Pair choosing** From all pairs of words (w_i, w_j) in w,
   (a) Suppose $w_i \ldots w_j$ is a new pattern, apply the transformation in Section 4.2 within each sentence;
   (b) Merge all appearances of (w_i, w_j) inside a sentence as a new word $w_i w_j$ in w to get a new word list $w^{i,j}$;
   (c) Based on $w^{i,j}$ we can get the updated dictionary $D^{i,j}$ — $D^{i,j}$ is built by first adding $w_i w_j$ into D and then remove w_i if there is no remaining appearance of single w_i in $w^{i,j}$ and finally remove single w_j if there is no remaining appearance of w_j in $w^{i,j}$;
   (d) Calculate $L(D^{i,j}) + L(x^n|D^{i,j})$, which equals to $L(D^{i,j}) + L(w^{i,j}|D^{i,j})$.
   The pair
   $$(w_a, w_b) = \arg\max_{w_i, w_j} L(D) + L(x^n|D) - L(D^{i,j}) - L(x^n|D^{i,j}) \ , \qquad (8)$$
   is chosen.
3. **Dictionary Update** If $L(D) + L(x^n|D) - L(D^{a,b}) - L(x^n|D^{a,b}) > 0$, which indicates by merging (w_a, w_b) we may achieve a further compression of description length, then $D \leftarrow D^{a,b}, w \leftarrow w^{a,b}$, and repeat **Pair Choosing** and **Dictionary Update**. Otherwise, the old D and corresponding segmentation w are returned as the output.

Fig. 1. Brief Algorithm for Building the Dictionary

In fact, in **Pair Choosing** what is really concerned is only the compressed length in bits and the whole transformation, merging stuff is unnecessary. To see this, let's first derive how many bits of compression we could achieve by merging w_a and w_b. For briefness of discussion, we assume that $w_a \neq w_b$. Let Ω be the number of words before merging; α be the number of w_a before merging; β be the number of w_b before merging; γ be the number of merged token pairs; and D' be the new dictionary after merging. According to (4) Before merging, the

data length is

$$L_D = \Omega \log \Omega - \sum_{w \in D} c(w) \log c(w) \ . \tag{9}$$

The new length after merging is

$$L_{D'} = (\Omega - \gamma) \log(\Omega - \gamma) - \sum_{w \in D'} c'(w) \log c'(w) \ . \tag{10}$$

The c' above is defined as

$$c'(w) = \begin{cases} c(w) & w \notin \{w_a, w_b, w_a w_b\} \ , \\ c(w) - \gamma & w \in \{w_a, w_b\} \ , \\ \gamma & w = w_a w_b \ . \end{cases} \tag{11}$$

The compression of data length is

$$\begin{aligned} \Delta_1 &= (\Omega \log \Omega - \sum_{w \in D} c(w) \log c(w)) - ((\Omega - \gamma) \log(\Omega - \gamma) - \sum_{w \in D'} c'(w) \log c'(w)) \\ &= \Omega \log \Omega - (\Omega - \gamma) \log(\Omega - \gamma) + (\alpha - \gamma) \log(\alpha - \gamma) + (\beta - \gamma) \log(\beta - \gamma) \\ &\quad + \gamma \log \gamma - \alpha \log \alpha - \beta \log \beta \end{aligned} \tag{12}$$

The compression of model length is relatively simple — recall the process of dictionary update — the changes from D to D' are

1. Adding a new item $w_a w_b$.
2. Removing w_a if w_a only appears as $w_a w_b$ or equivalently $\alpha = \gamma$.
3. Removing w_b if w_b only appears as $w_a w_b$ or equivalently $\beta = \gamma$.

The compression of model length is thus

$$\Delta_2 = \begin{cases} -(\lambda(w_a) + \lambda(w_b) + 1) \log(|X| + 1) & \alpha \neq \gamma, \beta \neq \gamma \\ \log(|X| + 1) & \alpha = \gamma, \beta = \gamma \\ -\lambda(w_b) \log(|X| + 1) & \alpha = \gamma, \beta \neq \gamma \\ -\lambda(w_a) \log(|X| + 1) & \alpha \neq \gamma, \beta = \gamma \end{cases} \tag{13}$$

As we can see, in **Pair Choosing** step, only the value of $\Delta_1 + \Delta_2$ is actually necessary rather than the updated word list and dictionary. A much more efficient algorithm could be obtained by directly calculating the compressed length and do all the expensive work including the transformation, updating the dictionary and merging words in data only after the pair is chosen.

However, such an algorithm has a much more vast search space than the continuous case, so usually some general heuristics are needed to speed up the computation. The following heuristics are used in our implementation,

– Divide learning into two phases: consider learning only continuous patterns in the first phase and learn non-continuous patterns based on that in the second phase;

– In the second phase,
  - Consider only non-adjacent pairs, assuming that all adjacent pairs is processed in the first phase;
  - Do not merge words with only near dependency, namely "a", "an", "the" or "in", "on", "for";
  - Assume that a non-continuous pattern pair must contain at least one word that is added into the dictionary from merging letters either in the continuous learning phase or the past non-continuous learning process.

4.4 Clustering Dictionary Items

Now we have a dictionary which holds all possible patterns. Since the dictionary discriminates two item explicitly even they only differ in one or two letters, it is helpful to cluster the items by their similarity in form. To do this, we first define a derivation relation $\prec$.

Definition 7. *For two strings a and b, $a \prec b$ if and only if $a \neq b$ and a is sub-sequence of b in terms of letters and $a \neq b$. We say b is a derivation from a if $a \prec b$.*

Next we consider only items longer than one letter, let E be the dictionary with one-letter items eliminated, then for each $a \in E$, we denote all its derivations as

$$\Phi_a = \{b | b \in E \wedge a \prec b\} \ . \tag{14}$$

It's easy to see that if a is a derivation of b then $\Phi_a \subset \Phi_b$. In addition, there exists a minimal cover P of $\{\Phi_a | a \in E\}$ such that $\forall a \in E,\ \exists \Phi_b \in P,\ \Phi_a \subset \Phi_b$. This P is the chosen as the cluster of patterns sorted by derivations. For example, suppose we have learned the following items,

– "the argument",
– "the argument is",
– "to strengthen the argument",
– "to strengthen".

The corresponding cluster should be {"the argument", "the argument is", "to strengthen the argument"} and {"to strengthen", "to strengthen the argument"}.

5 Experiment Results

5.1 Test Corpus and Platform

We test the algorithm on a corpus of GRE Analytical Writing essays with over 110000 words. The corpus is first preprocessed to omit all punctuations except the sentence ending ones, in order to avoid noise brought by commas and quotes. For faster convergence of the algorithms, each English word is considered a letter instead of real English letters. The whole algorithm is implemented in Python and run with Python 2.6.2 on a PC with a Intel Core Duo T2300E processor. Only one core is used in our experiment. It takes about 40 minutes to finish the first learning phase and about 200 minutes to finish the second learning phase.

Table 1. Top 4 Biggest Continuous Pattern Groups

(a) Patterns Derived from "the argument", Total Appearances: 588

Count	Pattern
255	the argument
62	the argument is
45	to strengthen the argument
35	in the argument
27	to better assess the argument I would need
23	problem with the argument involves
22	the argument relies on
17	better evaluate the argument we would need more information about
16	problem with the argument is that it
15	the argument assumes that
14	in conclusion the argument is unconvincing as it stands
13	the argument's proponent
11	to better assess the argument
10	better evaluate the argument I would need
8	better evaluate the argument we would need
8	problem with the argument is
6	the argument is unconvincing as it stands
1	problem with the argument lies

(b) Patterns Derived from "of the", Total Appearances: 542

Count	Pattern
459	of the
19	of the study
19	of the two
16	representative of the
11	percent of the
10	percentage of the
4	the long term viability of the
4	in terms of the

(c) Patterns Derived from "the author", Total Appearances: 520

Count	Pattern
357	the author
82	the author's
19	the editorial's author
19	the author must
16	the author cannot
16	the author provides no evidence that
11	the letter's author

(d) Patterns Derived from "in the", Total Appearances: 515

Count	Pattern
325	in the
48	in the first place
35	in the argument
20	in the future
19	in the area
14	in the past
14	in conclusion the argument is unconvincing as it stands
12	in the foreseeable future
10	in the coming year
5	in the marketplace
5	in the near future
4	enrolled in the
4	in terms of the

5.2 Results of Continuous Pattern Learning

By ranking all sets in the cluster, we can know about the usage information of a group of related continuous patterns. Table 1 lists top 4 biggest or most frequently used continuous pattern groups. The results fit human summary of the data from the critical writing textbooks very well, covering most of the human summary. A small problem is that "the" appears almost after each preposition such as "in" and "of". This problem is only observed with the three articles "a", "an" and "the" since their vast use, the model tends to consider them a part of the preceding word. A simple heuristic forbidding "the" to be the left word when merging may be added to avoid this. However, it turns out this does not affect the results significantly. Table 2 lists next 8 groups of patterns after pruning the noise of "the".

Another interesting fact is our algorithm have learned many rare collocations from the corpus and exhibiting excellently accurate phrase boundaries. Table 3 lists a random part of the phrases learned although they only appeared once or twice in the text.

5.3 Results of Non-Continuous Pattern Learning

The results of non-continuous pattern learning are not so prominent compared to continuous pattern learning but still acceptable. Table 4 lists top 40 biggest

Table 2. Next 8 Groups of Continuous Patterns After Pruning "the" Noise

(a) Patterns Derived from "it is", Total Appearances: 417

Count	Pattern
146	it is
125	it is entirely possible
75	it is possible
23	it is likely
17	it is impossible to
10	it is highly
8	it is unfair
7	it is equally possible
3	it is unreasonable
3	it is equally

(b) Patterns Derived from "number of", Total Appearances: 190

Count	Pattern
126	the number of
48	number of
13	the total number of
3	a disproportionate number of

(c) Patterns Derived from "this argument", Total Appearances: 179

Count	Pattern
87	this argument
49	this argument is
43	in this argument

(d) Patterns Derived from "does not", Total Appearances: 150

Count	Pattern
81	does not
12	does not mean
10	it does not follow
9	does not necessarily indicate
8	does not necessarily mean
8	does not indicate
8	does not state
7	does not necessarily
7	does not follow

(e) Patterns Derived from "the fact", Total Appearances: 141

Count	Pattern
90	the fact
30	the mere fact
21	cites the fact

(f) Patterns Derived from "fails to", Total Appearances: 136

Count	Pattern
45	fails to
28	fails to convince
15	fails to indicate
13	fails to provide
10	fails to account for
10	fails to consider
8	fails to take
7	fails to deliver on its premise

(g) Patterns Derived from "would need", Total Appearances: 133

Count	Pattern
27	to better assess the argument I would need
19	I would also need
17	better evaluate the argument we would need more information about
16	I would need
12	we would need
10	better evaluate the argument I would need
9	I would need to know
9	we would also need
8	better evaluate the argument we would need
5	would need
1	would also need

(h) Patterns Derived from "to support", Total Appearances: 136

Count	Pattern
48	to support
28	to support this recommendation
22	to support this
21	to support this conclusion
9	to support this claim

non-continuous pattern groups. Those patterns can be roughly divided into two classes — content-related patterns such as "the professor ... distant learning"; and content-independent patterns such as "without ruling out ... cannot convince me". In fact, the choice of corpus accounts for the such many appearances of content-related patterns — the GRE Analytical Writing has a relatively narrow topic pool. Even though, it is worth pointing out that our algorithm succeeds in learning many useful content-independent patterns, marked in bold. At the same time, it is worth pointing out that our transformation representation reserves long-distance dependencies in non-continuous patterns well, for example,

> ... accurately reflect donation rates **not only** to all religious environmental and educational institutions **but also** to the broader group of all charitable institutions ...

Moreover, it can be observed that even the most frequent non-continuous pattern group appears much less frequently compared to those continuous ones. In fact, such data sparseness is the major problem of non-continuous pattern

Table 3. Part of Patterns Learned in Spite of Few Appearances

- recyclable materials	- experience informs us	- delicate tissues
- vice versa	- is being threatened by	- new york
- with certainty	- retirement destination	- smaller nonprofit hospitals
- natural habitat	- donate substantial sums	- mesa operation
- costlier service	- torn down	- north america
- donations earmarked	- distinct unwanted	- mojave desert
- independently tested	- sophisticated scientific methods	- gulf shrimp

Table 4. Top 40 Biggest Non-Continuous Pattern Groups

Group	Count	Pattern
1	74	**the author ... in**
2	29	**it is entirely possible ... or**
	20	**it is possible ... or**
	18	**it is ... or**
3	36	the author ... is
4	34	such as ... or
5	33	the argument ... is
6	32	the writer ... in
7	28	would be ... in
8	26	...'s ... would be
9	25	**the number of ... is**
10	22	in ... the same
11	19	the fact ... in
12	18	this ... the case
13	17	**even if ... is**
14	17	**the arguer ... by**
15	17	that these ... are
16	16	**perhaps ... or perhaps**
17	15	**in summary ... based on**
18	14	**in conclusion ... is**
19	14	**to support this recommendation ... points out**
20	14	people who ... are
21	14	the president ... will
22	13	**it is ... possible**
23	13	in this memo ... its
24	13	**recommends ... in order to**
25	13	**or perhaps ... than**
26	13	the professor ... distance learning
27	12	**without ruling out ... cannot convince me**
28	12	the study ... it
29	9	**not only ... but also**
	3	**not only ... but**
30	12	1 ... 2 ... and 3
31	11	the manager ... ad lib
32	11	**to strengthen it ... must provide**
33	11	**however ... provides no evidence**
34	11	points out ... has
35	11	**to strengthen it ... must show**
36	11	residents ... public school education
37	10	curfew ... crime rate
38	10	bargain brand ... cereal
39	10	how many ... were
40	8	the president ... automate ... salaries
	2	the president ... automate

learning and the relative fewer usages of non-continuous patterns in actual English might as well accounts for this issue.

6 Conclusion

With the help of a dictionary based probabilistic language model, the problem of learning text patterns is formalized into uncovering the in-decomposability within patterns by treating them as a whole word, which fits well into the learning framework of MDL. The language model is very simple in mathematics, but it yields fairly promising results in continuous pattern learning. The transformation representation preserves long-distance dependencies in non-continuous patterns well. Experiment results for the non-continuous learning phase look less prominent than the continuous ones due to the sparseness of data, nevertheless the algorithm is still able to learn many content-independent patterns such as "not only ... but (also)". In general, the overall results have wide coverage when compared with human summary from the essay writing textbook. The major advantage of our approach is its totally unsupervised learning, which requires almost no prior knowledge about the corpus.

Some related research could be found in the study of multiword expressions (MWE) such as [10] and [11]. However, most current researchers concentrates on learning MWEs within a limited syntactical structure such verb phrases while our approach does not take structural features into consideration.

A possible improvement lies in the language model. Current model discriminates two patterns in spite of even a slight difference such as "I would need" and "I would also need". However, many similar patterns are worth considering not to be so discriminated, such as (1) "I would need" and "I would **also** need"; (2) "in the future" and "in the **near** future"; (3) "it is **possible**" and "it is **likely**". At the same time, it is not necessary to assume i.i.d. to gain exchangeability for learning non-continuous patterns. In fact there are many random processes with exchangeability and by introducing those models, the results might get improved.

Acknowledgments This research was funded by Peking University Education Foundation's Undergraduate Research Fellowship, and also it was supported by Beijing Natural Science Foundation (No. 4032013) and the 985 project of Peking University (No. 048SG/46810707-001). We would like to thank Dr. Zhan Weidong for his suggestions from the linguistic perspective and the anonymous reviewers for their comments.

References

1. Kushmerick, N., et al.: Wrapper induction: Efficiency and expressiveness. Artificial Intelligence **118**(1) (2000) 15–68
2. Soubboutin, M.M.: Patterns of potential answer expressions as clues to the right answers. NIST SPECIAL PUBLICATION SP (2002) 293–302
3. Ravichandran, D., Ittycheriah, A., Roukos, S.: Automatic derivation of surface text patterns for a maximum entropy based question answering system. In: Proceedings of the HLT-NAACL Conference. (2003)
4. Rissanen, J.: Stochastic complexity and modeling. The Annals of Statistics **14**(3) (September 1986) 1080–1100
5. Marcken, C.D.: The unsupervised acquisition of a lexicon from continuous speech. Citeseer (1995)
6. Goldsmith, J.: Unsupervised learning of the morphology of a natural language. Computational Linguistics **27**(2) (2001) 153–198
7. Grünwald, P., Rissanen, J.: The minimum description length principle. The MIT Press (2007)
8. Barron, A.R., Cover, T.M.: Minimum complexity density estimation. IEEE transactions on information theory **37**(4) (1991) 1034–1054
9. Shtar'kov, Y.M.: Universal sequential coding of single messages. Problemy Peredachi Informatsii **23**(3) (1987) 3–17
10. Boukobza, R., Rappoport, A.: Multi-word expression identification using sentence surface features. In: Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing, Singapore, Association for Computational Linguistics (August 2009) 468–477
11. Fazly, A., Cook, P., Stevenson, S.: Unsupervised type and token identification of idiomatic expressions. Computational Linguistics **35**(1) (2009) 61103