Unsupervised Lexical Learning As Inductive Inference via Compression

Chunyu Kit

Department of Chinese, Translation and Linguistics
City University of Hong Kong
Tat Chee Ave., Kowloon, Hong Kong
ctckit@cityu.edu.hk

Abstract

This paper presents a learning-via-compression approach to unsupervised acquisition of word forms with no *a priori* knowledge. Following the basic ideas in Solomonoff's theory of inductive inference and Rissanen's MDL framework, the learning is formulated as a process of inferring regularities, in the form of string patterns (i.e., words), from a given set of data. A segmentation algorithm is designed to segment each input utterance into a sequence of word candidates giving an optimal sum of description length gain (DLG). The learning model has a lexical refinement module to exploit this algorithm to derive finer-grained word candidates recursively until no more compression effect is available. Experimental results on an infant-directed speech corpus show that this approach reaches a state-of-art performance in terms of precision and recall of both words and word boundaries.

1 Introduction

Studies on lexical learning is concerned, in general, with how a learner exploits its innate learning mechanisms, existing knowledge, if any, and other available facilities, like supervision of various kinds, to acquire more knowledge about words – the very basic units in a language to make up utterances for human communication. The most essential lexical knowledge is word forms, with which some syntactic and semantic properties (e.g., part of speech, meaning) are associated so as to make our utterances meaningful. In this sense, word forms are the carrier of meaning. Therefore, a lexical learner must have some means at the very beginning of lexical learning to infer the word forms before they can associate any meanings with them.

In this research we take a computational approach to study the lexical acquisition problem with a focus on unsupervised learning of word forms, involving no other lexical (such as syntactic and semantic) properties. Unsupervised learning assumes no prior knowledge at the starting point of the learning and no supervision available during the learning. Different approaches assume the learner to have different initial ability, i.e., the innate learning mechanisms. A very interesting assumption for this study is that the learner has an initial mechanism to do little more than string counting. It does not know there are words to learn. And it does not "learn", in a sense, but just attempts to derive a least-cost representation for the input data in terms of the string counts, each of which gives an indication of how many bits can be saved via extracting a string as a lexical item.

This minimal initial ability is interesting. It follows, in a sense, Chomsky's thoughts on the minimality of grammar for natural language and his arguments for the minimally necessary

innate structure (and ability) for language learning faculty, scattered in his linguistic theories from [5] to [6]. One may take is as a piece of evidence against his assumption of a powerful universal grammar, if unsupervised learning starting from this initial point could succeed. Our purpose here, however, is simply to demonstrate the power of our learning approach: a counting machine can learn words to a great extent of success. The minimality of initial ability is important in that between two learning approaches achieving a similar learning performance, the one with less initial ability (and knowledge) indicates a more effective learning.

The theoretical inspiration for this research comes from *algorithmic information* (or *Kolmogorov complexity*) theory [24, 13, 4, 14], including (1) Solomonoff's *inductive inference* theory [24] – the first of the three origins of the algorithmic information theory, (2) the MDL and MML principles by Rissanen [20, 21, 22] and by Wallace and colleagues [29, 30], respectively[1], (3) Vitányi and Li's formulation of the *ideal* MDL in terms of Kolmogorov complexity [27, 28] and, in particular, their idea (or "intuition", in their own terms) on how to conduct inductive inference via compression on a given data set by squeezing out the embedded regularities piece by piece [14] (p.351). In a sense, the work reported here can be thought of as an attempt to formulate and implement this nontrivial "intuition".

An important theme in the studies of learning is to study how to trace the underlying machinery that has generated the data, via detecting the regularities in the data. Compression is argued to be an effective approach for an optimal approximation to the generally non-computable Kolmogorov complexity over a given data set [28, 27]. In the context of lexical learning, the data set consists of utterances, each of which is a sequence of atomic symbols in the language in question, i.e., phonemes in sound or letters in script. Ideally, learning from a data set is to retrieve an achievable minimal representation for the data, extracting all regularities from the data, and the final result, consequently, cannot be further compressed. Since this minimal representation is not reachable in general, the best we can do is to compress the data as much as possible under some constraints, e.g., the ones imposed by the representation format allowable in the learning.

Following Solomonoff's insight into the duality of compression and regularities [24], i.e., anything that can compress the data is a piece of regularity and any regularity can (be used to) compress the data, we may state that a model that can compress the data to a greater extent is a better model in general, in the sense that it captures more regularities in the data and thus reaches closer to the true machinery that has generated the data. In this sense, unsupervised learning is a process of inductive inference to derive the optimal set of regularities from the data that *can* compress the data the most. Notice, however, that it is a different issue whether the regularities so learnt are to be applied to carry out the compression. In this learning-via-compression approach to lexical learning, the compression is more a way of thinking about the computation involved in the learning process than a real procedure for compressing the input data.

As in other more cognition oriented studies on language learning, we also adopt the assumption that the lexical learning follows the *least-effort principle* [31] to learn from the natural language data generated by human language behaviors that are observed to be governed by the least-effort principle in language production. However, instead of interpreting the effort

[1]The subtle difference between MDL and MML is not critical to our research here. A lengthy discussion on the difference of the two can be found in a special issue of *The Computer Journal* (Vol.42, No.4, 1999) on Kolmogorov complexity. More important to our research is the undelrying philosophy they share. Thus, the abbreviation MDL is assumed to subsume both within this paper, for the sake of simplicity.

as the energy consumed by the learning process, we think of it as the cost in terms of the number of bits in the representation for the data. This point is critical, because from the point of view of a learning-via-compression approach, learning is merely the process to derive an economic representation for the data, and the regularities (e.g., string patterns) so resulted are both the by-products, in a sense, of this derivation and the means towards a more compact representation.

Computational studies on lexical learning fall into different categories, e.g., *connectionist* (or *neural network*) approaches, genetic algorithms and *probabilistic* models. Many probabilistic models adopt, in one way or another, the basic idea of learning-via-compression and the MDL principle. Representative ones of these models include the *word grammar* by Olivier [18] – a noticeable piece of early work on lexical learning, the *distributional regularity* (DR) model by Brent and Cartwright [3], the *concatenative* model by de Marcken [8, 9], and Brent's probabilistically-sound model and its implementation – the MBDP-1 system [2], among many others. Olivier's work demonstrated the formulation of lexical learning as an optimisation process in terms of an objective function with dynamic programming techniques. The work of de Marcken's outputs impressive tree structures, most of which appear to be consistent to morphological structures, but only about two out of a dozen are real words, rather close to the recall of the random baseline used in Brent's work [3, 2]. Brent's MBDP-1 system gives a learning performance with a balanced precision and recall, both above 70%, demonstrating the state of the art of computational studies specifically devoted to lexical learning. Venkataraman's recent work [26] shows that this performance can be achieved by available n-gram language modelling, and the learning curves also look highly similar. Our leaning model presented here achieves a slightly better performance, and our approach appears significantly simpler, most likely due to our straightforward formulation of the idea of learning via compression within the MDL framework.

In this paper we report on our recent work in unsupervised lexical learning via compression to derive a least-effort representation within the theoretical framework of inductive inference following the MDL principle. The purpose of the research on lexical learning is multi-fold. First, it aims to test the hypothesis that there is a mechanism underlying language acquisition that seeks for the least-effort representation for the input data. Secondly, it explores machine intelligence with a focus on examining how much a computer can learn from natural language data given that it has only a minimum innate capacity, that is, the capacity to differentiate between signals or, equivalently, characters in texts, together with other related capacities derived from this basic capacity, such as counting distinct signals and strings in a given corpus. It is remarkable to demonstrate that a counting machine can learn words from natural language data with little prior knowledge and supervision. However, what is more important to show is not that such success is resulted form the power of the learning mechanism involved. Instead, it is that linguistic regularities in real language data can be captured statistically and information-theoretically by a counting machine. Thirdly, the research is expected to lead to a better understanding of the nature of the human language acquisition device that is assumed to have a minimally necessary innate structure and ability, in the hope that the research on machine learning of natural language, especially when a minimum innate capacity is assumed, can shed light on the mechanism of human language acquisition: if machine learning indicates that linguistic regularities embedded in language data play a critical role in facilitating language acquisition in addition to the innate mechanisms, it is understandable that they may play a

similar role to enable human infants to learn a language so easily.

The rest of the paper is organized as follows. Section 2 formulates the goodness measure for the compression effect of extracting a substring as a word candidate from a given set of data. Section 3 presents a Viterbi algorithm to give an optimal segmentation for each input utterance in terms of the goodness measure, and Section 4 a three-phase lexical learning model based on this algorithm. In Section 5 we present testing data for evaluation and corresponding learning results output from the learner, including word clumps as intermediate results showing how the learner works towards finer-grained lexical items. In Section 6, we define a number of measures, including word, word boundary and word type precision and recall, as a comprehensive evaluation for the learning performance, and report the evaluation results in terms of these measures. A number of interesting problems encountered in the learning are discussed in Section 7, before we conclude our work in Section 8.

2 Goodness Measure

When a learning problem is formulated as an optimisation problem, an objective function is needed to guide the learning. In order to enable our unsupervised learning approach through compression to carry out the optimisation, a goodness measure is required to evaluate the benefit from extracting each possible word candidate from the input corpus and putting it into the lexicon.

A such goodness measure, termed *description length gain* (DLG), was formulated in [12] and [10] to compute the compression effect of this kind. Its formulation is as the following: given a corpus $X = x_1 x_2 \cdots x_n$ as a sequence of linguistic tokens (e.g., characters in our case), the DLG from extracting a subsequence $x_i x_{i+1} ... x_j$ (also denoted as $x_{i..j}$) $(i < j)$, from X as a rule in the form $r \rightarrow x_{i..j}$ is defined as

$$DLG(x_{i..j} \in X) = DL(X) - DL(X[r \rightarrow x_{i..j}] \oplus x_{i..j}) \tag{1}$$

where $X[r \rightarrow x_{i..j}]$ represents the resultant corpus from the operation of replacing *all* instances of $x_{i..j}$ with the new symbol r throughout X, and $\oplus$ denotes a string concatenation operation with a delimiter inserted in between, and $DL(\cdot)$ is the empirical description length that can be estimated by the Shannon-Fano code or Huffman code as below, following classic information theory [23, 7].

$$\begin{aligned} DL(X) &= |X| \, \hat{H}(X) \\ &= -|X| \sum_{x \in V} \hat{p}(x) \log_2 \hat{p}(x) \\ &= -\sum_{x \in V} c(x) \log_2 \frac{c(x)}{|X|} \end{aligned} \tag{2}$$

where $|\cdot|$ denotes the length of a corpus, $c(\cdot)$ the frequency of a token in the corpus, and $\hat{p}(\cdot)$ the relative frequency that is conventionally estimated as $\frac{c(\cdot)}{|X|}$. Straightforwardly, the average DLG of $x_{i..j}$, i.e., the compression effect of extracting *each* individual instance of $x_{i..j}$, is

$$DLG_{\text{av}}(x_{i..j} \in X) = \frac{DLG(x_{i..j} \in X)}{c(x_{i..j})} \tag{3}$$

A significant benefit from the above formulation is that we need not carryout the transformation to compute DL for the new corpus $X' = X[r \rightarrow x_{i..j}] \oplus x_{i..j}$. Following (2), it can be formulated straightforwardly as below in (4), using the new count $c'(x)$ in the new corpus X'.

$$DL(X') = -\sum_{x \in V \cup \{r, \oplus\}} c'(x) \log_2 \frac{c'(x)}{|X'|} \tag{4}$$

where $c'(\cdot)$ is a token count in X'. The focus of this computation is thus on how to derive $c'(\cdot)$ in the new corpus X' without deriving X'. Notice that in order to derive X' we must carry out the costy transformation by the extraction, replacement and concatenation. This is what we do not want. Instead, we prefer to drive $c'(x)$ for any x directly from the known counts $c(x)$ and $c(x_{i..j})$ in the original corpus X. $c(x_{i..j})$ is the count of an ngram of arbitrary length.

Actually, this derivation is quite straightforward, as given below in (5), where $c(\cdot)$ and $c(\cdot \in x_{i..j})$ denote an n-gram count in X and in $x_{i..j}$, respectively.

$$c'(x) = \begin{cases} c(x_{i..j}) & \text{if } \ x = r; \\ c(\oplus) + 1 & \text{if } \ x = \oplus; \\ c(x) & \text{if } x \notin x_{i..j}; \\ c(x) - c(x_{i..j})c(x \in x_{i..j}) + c(x \in x_{i..j}) & \text{if } x \in x_{i..j}. \end{cases} \tag{5}$$

The first three cases are trivially simple. In the last case, i.e., the general case, the second item is the number of x's reduced by the extraction of $x_{i..j}$ and the third item is the number of x's remaining in the only instance of $x_{i..j}$ in the model part. Consequently, the length of the updated corpus after the transformation for extracting $x_{i..j}$ is

$$|X'| = |X| - c(x_{i..j})|x_{i..j}| + |x_{i..j}| + 1 \tag{6}$$

where the second item on the right-hand side is the length reduced by the extraction, the third item is the length of the extracted pattern $x_{i..j}$, which is concatenated to the updated corpus, and 1 is for the delimiter introduced by the concatenation.

Since all fragments of the input can be a word candidate. The learning needs to examine all of them, i.e., all n-gram items, in the corpus. Although n-grams of arbitrary lengths in a large-scale corpus are known to be huge in number, we have developed the *Virtual Corpus* (VC) system [11], based on the *suffix array* data structure [17], as a fairly efficient approach to handling them, including counting, storing and retrieval.

3 Algorithm

With the aid of the DLG formulated above, the unsupervised lexical learning becomes an optimal segmentation problem seeking for the sequence of word candidates over an input utterance with the greatest sum of DLGs. Given an utterance $U = t_1 t_2 \cdots t_n$ as a string of some linguistic tokens (either characters, phonemes or syllables) in a given corpus C, the optimal segmentation $OS(U)$ over U such that the sum of DLGs over the word candidates is maximal can be formulated as below[2]

$$OS(U) = \underset{s_1 \cdots s_k \ s.t. \ U = s_1 \oplus \cdots \oplus s_k}{\arg\max} \sum_{i=1}^{k} DLG_{\text{av}}(s_i) \tag{7}$$

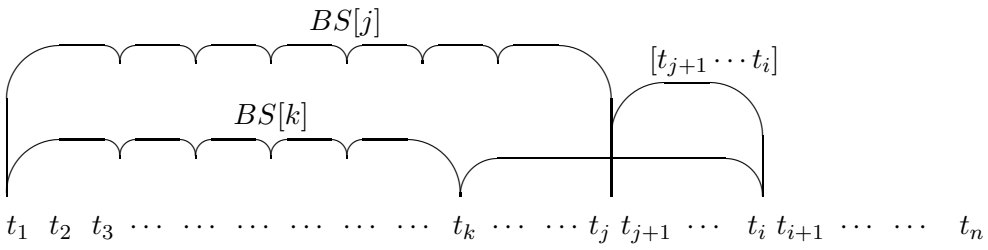

Figure 1: An illustration for the Viterbi segmentation

for $0 < k < |U|$.

Accordingly, an algorithm to implement this optimization is formulated with the aid of dynamic programming, following the basic idea of Viterbi algorithm It uses a list of intermediate variables $BS[i]$ (for $i = 1, 2, \cdots, n$) to store the best segmentation over $t_1 t_2 \cdots t_i$. A segmentation is a list (or chain) of adjacent segments (i.e., word candidates). The DLG over a list of segments, e.g., $DLG(BS[j])$, is the sum of the DLG of the individual segments in the list, as defined in (8).

$$DLG(BS[j]) \doteq \sum_{s \in BS[j]} DLG(s) \tag{8}$$

Following the illustration in Figure 1, the optimal segmentation (OS) algorithm is as straightforward as follows, with $\uplus$ denoting the operation of joining two lists.

1. Starting from $i = 1$,
2. Once $BS[i-1]$ is derived, let $BS[i] = BS[j] \uplus \{[t_{j+1} \cdots t_i]\}$ for
$$j = \arg\max_{k<i} DLG(BS[k] \uplus \{[t_{k+1} \cdots t_i]\}) \tag{9}$$
3. Repeat the above step until $i = n$.

To speed up this algorithm, we can set an empirical condition $c([\ t_k\ \cdots\ t_i]) > f_{min}$ (usually, 1) for the search at each step, to lead the algorithm to avoid fruitless iterations on strings with a two low frequency, in particular, 1. Notice that all strings with a count $c = 1$ have a negative DLG, and they are all long strings that can be (or have been) broken into shorter ones with a positive DLG.

4 Learning model

The learning model for unsupervised lexical learning exploiting the above Viterbi segmentation consists of two phases, as depicted in Figure 2, each of which involves an application of the algorithm to infer lexical units at different granularity. Another phase is word segmentation using the lexicon resulted from the learning. It is post-learning application of the segmentation algorithm to identify individual words for later process of language understanding.

[2]We denote $DLG(s_i \in C)$ and $DLG_{\text{av}}(s_i \in C)$ as $DLG(s_i)$ and $DLG_{\text{av}}(s_i)$, respectively, in discussion concerning only a default corpus C, for the sake of simplicity.

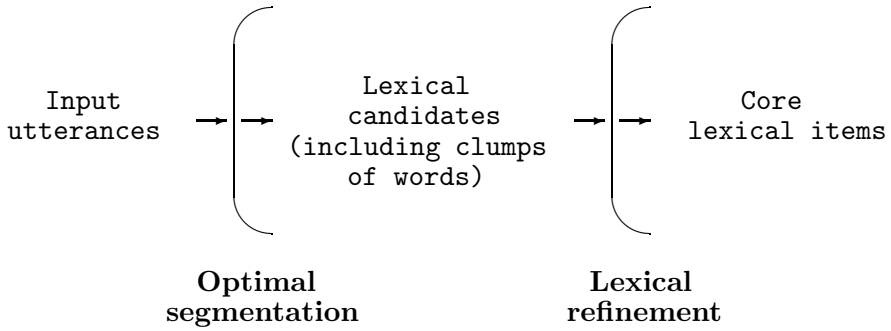

Figure 2: The model of lexical learning behind our lexical learning algorithms

1. Induction of lexical candidates by optimal segmentation on input utterances,
2. Lexical refinement by optimal segmentation on individual lexical candidates, and
3. Word segmentation by optimal segmentation using the lexicon acquired.

The optimal segmentation algorithm is the underlying mechanism supporting all the three under different circumstances of learning. The first two constitute the process of the lexical learning for word form discovery, and the last step uses the learned lexicon to perform word segmentation for language understanding. A rationale beneath this learning model is that it would be redundant to have distinct cognitive mechanisms for word discovery and word segmentation. The word segmentation is regarded as a special case of word discovery to determining word forms with a lexicon that is presumed to be adequate. Cognitively, once out-of-vocabulary words are encountered during word segmentation, the mechanism for word discovery will be invoked to infer unseen words. In our experiments, we get the final results from the last phase for the evaluation of the learning performance.

For the purpose of exploring human lexical learning mechanisms with this computational approach, it is reasonable to assume that only those n-gram items containing at least a vowel can be a word candidate. It is known that syllables are the basic units of speech representation and every word must contain at least a syllable, that is, a word contains at least a vowel (character). It is rather straightforward to implement this constraint upon the optimal segmentation process: every chunk in the segmentation must contain at least a vowel character. From now on we will denote the optimal segmentation with the vowel constraint as `OS+V` and the one without as `OS`. In the case that we specify that the apostrophe `[']` is an equivalent to a vowel character, in addition to `[aeiouy]`, we denote the algorithm as `OS+V'`. Clearly, it is a less constrained version of `OS+V`. This specification is to enable the learning algorithm to recognise the bound morphemes like `[-n't]` and `['ll]` as individual lexical items in English. Without this specification, the `OS+V` algorithm will have no chance to show its ability to learn these morphemes, because it always has to adhere them to lexical items with a vowel under the vowel constraint.

We can also iterate these two algorithms, in the essence of the EM algorithm, to test their learning capacity – we will see how far they can go in lexical learning on their own. The implementation is very simple: repeat each of the algorithms again and again and update the frequency of the n-grams according to the segmentation result in each iteration. It can be

expected that the learning performance will be improved iteration by iteration, until there is no more improvement, then we stop. We refer to these algorithms as `OS+EM`, `OS+V+EM` and `OS+V'+EM`, and the experimental results of these algorithms will be presented in the next section.

5 Testing Data and Learning Results

In this section we will present the input data to our lexical learner for testing and the learning results it outputs. It is based one this output that the learner's performance is evaluated. We will first give some rationales for selecting a text corpus of child-directed speech transcription as testing data, and then present the details of data preparation, with samples of the input and output and intermediate results of learning. The intermediate results also show how the learner works at each phase of learning.

5.1 Input Data

It is understood that written text corpora like the Brown corpus and the PTB corpus are not appropriate for testing an unsupervised lexical learning approach that is aimed at exploring the language-learning infants' lexical learning mechanisms. Instead, we have to use the language data that the children really receive in their language-learning environments. The CHILDES database [16, 15] is a collection of such data, contributed by a large number of scholars in the field of language acquisition. The Bernstein corpus [1], a naturally-occurring child-directed speech corpus, from the CHILDES collection is the most suitable data set for the purpose of testing our lexical learner's performance.

Another reason for choosing the Bernstein corpus, instead of other corpus from CHILDES, is that we intend to compare our work with the state-of-the-art approach that Brent has recently reported in [2], where the Bernstein corpus was used as testing data. The input data will be illustrated below in detail with examples. It is a corpus of plain text transcribed from child-directed speech. Figure 3 presents an exemplar fragment of the original transcription of a mother's speech to language-learning children in the Bernstein corpus from the CHILDES collection.

However, this is not yet the input for our lexical learner. We have to do some pre-processing to filter out the noise in the data. First of all, it is necessary to filter out the non-speech content in the data, including commentary notes, punctuation marks[3], etc. Next, we convert all capital letters into lowercase ones, except the initial letters in proper names, to prevent the learner from making an unnecessary distinction between a word and its capitalised version. Also, we add a special end-of-utterance symbol "`#`" to each utterance, to tell the learner where an utterance ends. This symbol is not part of the data on which the learner will perform the learning. It is necessary because we intend to feed the data to the learner utterance by utterance. After these steps, we have a text corpus consisting of a list of utterances, each of which is a sequence of characters ended by "#". Below in Figure 4 is the counterpart of the data in Figure 3 after the pre-processing.

This text corpus contains spaces as word delimiters between words. In order to test the

[3]The only exception is apostrophe `[']`. It is not removed from the input corpus, because it is used to represent a reduced vowel in bound morphemes in English orthographic texts, e.g., `[-n't]` and `[-'re]`. In order to enable lexical learner with a vowel constraint to detect individual bound morphemes, we have to inform the learner that an apostrophe is an equivalent vowel character. See later sections for more discussion.

```
*MOT: She's really into books right now.
*MOT: You want to see the book?
*MOT: Oh, look there's a boy with his hat.
*MOT: And a doggie.
*MOT: Oh, you want to look at this?
*MOT: W' look at this?
*MOT: Have a drink.
*MOT: OK now.
*MOT: Oh what's this?
*MOT: What's that?
*MOT: Get it.
*MOT: Get it.
*MOT: Get it.
*MOT: Is that for the doggie?
*MOT: Can you feed it to the doggie?
*MOT: Feed it +...
*MOT: Oh # put it in OK.
*MOT: OK.
*MOT: What are you gonna do?
*MOT: I'll let her <play with these>
      [//] play with this for a while.
```

Figure 3: A fragment of the original transcription of a mother's speech from the Bernstein corpus of the CHILDES collection

```
she's really into books right now #
you want to see the book #
oh look there's a boy with his hat #
and a doggie #
oh you want to look at this #
w' look at this #
have a drink #
ok now #
oh what's this #
what's that #
get it #
get it #
get it #
is that for the doggie #
can you feed it to the doggie #
feed it #
oh # put it in ok #
ok #
what are you gonna do #
i'll let her play with this for a while #
```

Figure 4: A fragment of the input corpus after pre-processing to filter out non-speech content

```
she'sreallyintobooksrightnow#          getit#
youwanttoseethebook#                   getit#
ohlookthere'saboywithhishat#           getit#
andadoggie#                            isthatforthedoggie#
ohyouwanttolookatthis#                 canyoufeedittothedoggie#
w'lookatthis#                          feedit#
haveadrink#                            oh#putitinok#
oknow#                                 ok#
ohwhat'sthis#                          whatareyougonnado#
what'sthat#                            i'llletherplaywiththisforawhile#
```

Figure 5: A fragment of the input corpus with spaces deleted

learning ability of an unsupervised learning algorithm, we have to make such word delimiters unnoticeable to the learner. A good way to hide the spaces from the learner is to delete them from the input data, as in Figure 5. The spaces are artificial delimiters in written texts, in the sense that there is nothing in continuous speech corresponding to such spaces. Removing them from the input corpus appears to be the right thing to do for the purpose of testing a learner's ability of learning the orthographic transcription of speech. A spaceless text such as the one in Figure 5 is also known as an unsegmented text; its counterpart with spaces is accordingly called a segmented text. We will use an unsegmented corpus of child-directed speech to test our lexical learner's performance. It is not easy for a human speaker to read off the words in an unsegmented text as the one in Figure 5. Without a certain learning capacity an unsupervised learner would not be able to infer words from a corpus of this type with an acceptable degree of success.

A question one may ask about the testing data is, why use orthographic text as input, not speech input? In order to answer this question, several points need to be clarified. First, we do not really need the speech input in the form of sound waves as input data for out study. We know that human infants' categorical speech perception turns the speech signals they receive in a sequence of sounds, known as phones, for each utterance. Our research is aimed at exploring the learning mechanisms (or strategies) that the language-learning infants may exploit to map the sound sequences to lexical items at the early stage of lexical learning when they have no knowledge about words in their languages. Involvement of too much speech processing details would not help highlight the purpose of our study. Furthermore, for technical reasons, all speech data suitable for computational studies of language learning are actually in text format as some kind of transcript, e.g., phonetic transcript.

So why don't we use phonetic transcripts as input? The only reason is the unavailability. If they were available, we would use them for the test with no hesitation. However, we are also happy with testing on the orthographic transcripts of the same corpus, for the following reasons. First, our learning approach is aimed at exploring the general learning mechanisms that human infants may use for learning words from language data, no matter what format the data is in and what distributional regularities that data may have. We are not interested in a learning mechanism that can deal with input data in one format but not others or in one language but not others. We believe all human infants use similar learning mechanisms, regardless of the

language involved, to deal with the lexical learning problem at the initial stage of language acquisition.

Second, our learning approach works through detecting regularities in the input data. Language data from different languages exhibit different regularities, and no matter what regularities there are in the input, our learning approach must be able to capture them; otherwise, it is not really a general approach. We know that both the phonetic and orthographic transcripts are transcribed from the same speech data, and that most regularities in the original speech data are well maintained in both types of transcript, in different form though. Furthermore, in comparison with an artificial phonetic transcription scheme, the orthographic transcript is more natural because it has evolved for thousands of years. The orthographic transcript carries its own inconsistency with the speech data and its own irregularities (e.g., \ə\ appears as `-er`, `-or`, `-ur`, `-ir` or in some other forms in orthographic texts). All these provide more challenges to, and therefore are a more creditable test for, our learning algorithms.

The orthographic transcripts of the Bernstein corpus have to undergo some necessary pre-processing before being input to the lexical learner as testing data. The main purpose of the pre-processing is to filter out the non-speech content from the corpus and do some necessary adjustment. We follow an essential principle in the data pre-processing, that is, we only do the *minimally necessary* adjustment.

The job of pre-processing that we have carried out includes the following aspects:

- Filtering out the non-speech content, including punctuation marks and commentary notes in square brackets;
- Converting capitalised words, except for proper nouns, to lowercase ones;
- Adjusting some special forms of words or abbreviated words (e.g., the abbreviated form of negation) back to standard orthographic forms, e.g., "`i (the)m`" → "`i am`", " `wouldn (i)t`" → "`wouldn't`".

Notably, the speech-related marking symbols, mainly braces and colons inside words, e.g., as in `(a)n(d)`, `beau:ti:ful`, `fa:v(o)rite` and `tel:e:phone` are all filtered out. Table 1 summarises a number of problems in the original corpus and the adjustments that have been carried out by the pre-processing process.

However, many non-standard word forms remain in the data, e.g., `w'` (we), `y'` (you), `ya` (you), `whatcha` (what do you) and `whatchya` (what do you). This kind of inconsistency really plays a critical role in testing the learner's learning ability.

Also, we allow one non-speech symbol, "`+`", to remain in the data, as in `peek+a+boo` and `bye+bye`, because it reflects the corpus constructors' intention that these strings should be considered as individual words or word-like compounds instead of as several words. And the unknown words in the form `xxx`, incomplete words such as `wh`, onomatopoeia (e.g., `woof woof woof`) and interjections (e.g., `oh`, `ooh` and `uh`) are also kept, contrary to how Brent prepared his testing data. If all these irregularities were cleaned up, the lexical learner would have a better performance.

The entire testing corpus of child-directed speech extracted from the Bernstein corpus is of 9702 utterances, 35K words and 143K characters. The average utterance length is 14.7 characters and 3.6 words; the average word length is 4.1 characters.

Problem	Example	Freq.	Adjustment	Result
:	dog:gie	6	Removal	doggie
	kit:ties	6		kitties
	kit:ty	4		kitty
()	(a)bout	18	Removal	about
	(a)n(d)	15		and
	an(d)	12		and
	d(o)	34		d
	(doe)s	8		does
	goin(g)	3		going
	(i)s	29		is
	(i)t's	24		it's
	(it')s	4		it's
	(it)'s	8		it's
	(o)k	14		ok
	(o)kay	2		okay
	(th)at	6		that
	(th)em	37		them
	wan(t)	26		want
	y(ou)	11		you
() + '	(doe)'s	1	Removal	does
	(i)'s	5		is
	wha'(t)s	1		what's
	wha(t)'s	2		what's
-n (i)t	aren (i)t	16	Transformation	aren't
	can (i)t	22	[-n (i)t] → [-n't]	can't
	couldn (i)t	2		couldn't
	doesn (i)t	42		doesn't
	don (i)t	151		don't
	don (i)tcha	2		don'tcya
	don (i)tchya	1		don'tchya
	hasn (i)t	1		hasn't
	haven (i)t	5		haven't
	isn (i)t	17		isn't
	mustn (i)t	1		mustn't
	shouldn (i)t	1		shouldn't
	wasn (i)t	1		wasn't
	won (i)t	5		won't
	wouldn (i)t	2		wouldn't
(the)m	i (the)m	86	Transformation [(the)m] → [am]	i am

Table 1: A summary of problems in the orthographic transcripts of the Bernstein corpus and their adjustment in the pre-processing

```
[she's][really][int][o][book][sright][now]
[youwantto][seethe][book]
[ohlook][there's][abo][ywith][his][hat]
[and][a][doggie]
[oh][youwantto][lookatthis]
[w'][lookatthis]
[havea][drink]
[o][know]
[oh][what'sthis]
[what'sthat]
```

```
[getit]
[getit]
[getit]
[isthat][for][thedoggie]
[canyou][feeditto][thedoggie]
[feed][it]
[oh][putitin][ok]
[ok]
[whatareyou][gonnado]
[i'll][le][ther][playwiththe][se]
[playwithth][isfor][a][whi][le]
```

Figure 6: A fragment of the output from the optimal segmentation of the Bernstein corpus

5.2 Learning Result

The unsupervised lexical learning is aimed at acquiring lexical forms from speech data where word boundaries are not marked by any means. The expected outcome from the learning algorithms is a lexicon consisting of a list of individual word forms. This representation for both the intermediate and final results of the learning in our research is in fact a deterministic regular grammar. In this grammar, the right-hand sides of rules are lexical candidates that are represented plainly as strings of the atomic symbols in the input corpus. The left-hand sides, which merely function as indices for the lexical candidates, are skipped in the representation, because the positions of the candidates in the lexicon play the same role as the skipped indices. The choice of skipping the left-hand sides in the lexicon also reflects the principle of simplicity – the underlying philosophy of our learning approach. Recall that our learning approach seeks for a lexicon as the simplest representation for the input data.

Here we will illustrate the representation formalism with fragments of the real output from our lexical learning experiments. The learning algorithms yyielding such output will be formulated in the next section. The output from the first step of the lexical learning – an optimal segmentation of input utterances into lexical candidates in terms of the DLG measure – consists of two parts: One is the results of the optimal segmentation on the input utterances, such as in Figure 6; and the other is the correspondent lexical candidates resulting from the segmentation, such as the ones in Table 2, where each candidate is has its count (or frequency), length, coverage and average DLG attached. The spacing between characters is for readability, and the symbol "+" is originally in the corpus. Many of the infrequent candidates in this lexicon, not shown here, are non-words; and most of the frequent ones, as shown in the figure, are real words or clumps of real words.

Therefore it is necessary to have a lexical refinement process in the lexical learning to turn the word clumps into individual real words. Table 3 gives a number of decompositions of word-clump lexical candidates into words (and other shorter clumps) during the lexical refinement process, where the right-most column is the DLG of each decomposition. Table 4 is the finer-grained lexicon that is output from the lexical refinement.

The final results of the lexical learning, i.e., the output from the optimal segmentation of the input utterances using the refined lexicon (obtained by the first two steps of the learning,

```
 57  6  342  12.6261 [ p r e t t y ]
 44  8  352  15.8121 [ t h o s e a r e ]
 30 12  360  33.3633 [ c l o s e t h e d o o r ]
 52  7  364  12.9680 [ t h i s o n e ]
 73  5  365  12.5871 [ m o m m y ]
 73  5  365   7.4265 [ h e l l o ]
 37 10  370  25.6890 [ w h e r e ' s t h e ]
 53  7  371  24.5830 [ b y e + b y e ]
 62  6  372  14.0373 [ y o u c a n ]
188  2  376  -3.4687 [ i t ]
 67  6  402  14.2962 [ d o g g i e ]
 41 10  410  23.8961 [ w h a t i s t h a t ]
207  2  414  -1.8734 [ o k ]
 52  8  416  18.1429 [ w h a t i s i t ]
 85  5  425   8.4561 [ i t ' s a ]
 43 10  430  25.5103 [ l o o k a t t h i s ]
108  4  432   6.5355 [ b o o k ]
 63  7  441  15.1237 [ t h e r e ' s ]
 37 12  444  30.8000 [ w h a t a r e t h o s e ]
 76  6  456  12.7278 [ w h a t ' s ]
152  3  456  -1.0154 [ t h e ]
477  1  477  -6.0098 [ a ]
161  3  483  -0.3685 [ s e e ]
 81  6  486   9.8742 [ i s t h a t ]
122  4  488   3.5881 [ t h i s ]
 60  9  540  23.9726 [ t h e d r a g o n ]
 98  6  588  14.9844 [ c a n y o u ]
197  3  591   2.2711 [ y o u ]
 74  8  592  22.1664 [ a l l r i g h t ]
 66  9  594  24.5681 [ t h e d o g g i e ]
149  4  596   3.7548 [ h e r e ]
161  4  644   6.1727 [ o k a y ]
 65 10  650  41.8614 [ p e e k + a + b o o ]
222  3  666   2.3890 [ a n d ]
 62 11  682  31.9754 [ t h a t ' s r i g h t ]
122  6  732  11.9507 [ t h a t ' s ]
110  7  770  15.4102 [ t h a t ' s a ]
157  5  785   7.2517 [ t h e r e ]
198  4  792   3.1169 [ t h a t ]
203  4  812   6.4316 [ l o o k ]
452  2  904  -2.7021 [ o h ]
230  4  920   5.0676 [ w h a t ]
329  4 1316   5.3064 [ y e a h ]
139 10 1390  29.2903 [ w h a t ' s t h i s ]
247 10 2470  29.1359 [ w h a t ' s t h a t ]
```

Table 2: Sample lexical candidates output from the optimal segmentation, in ascending order of coverage (= count × length). Each candidate's count, length, coverage and average DLG are attached on the left

```
Round 1 (The first 12 in 503 decompositions):
[ w h a t ' s t h a t ]     => [ w h a t ' s ] [ t h a t ]                15.8447
[ w h a t ' s t h i s ]     => [ w h a t ' s ] [ t h i s ]                16.3159
[ t h a t ' s r i g h t ]   => [ t h a t ] [ ' s ] [ r i g h t ]           8.8198
[ a l l r i g h t ]         => [ a l l ] [ r i g h t ]                      8.4251
[ c a n y o u ]            => [ c a n ] [ y o u ]                         3.3597
[ t h e d r a g o n ]      => [ t h e ] [ d r a g o n ]                  11.5103
[ w h a t a r e t h o s e ] => [ w h a t ] [ a r e t h o s e ]          14.4629
[ t h e r e ' s ]          => [ t h e r e ] [ ' s ]                       4.8175
[ l o o k a t t h i s ]    => [ l o o k ] [ a t ] [ t h i s ]             5.2867
[ w h a t i s i t ]        => [ w h a t ] [ i s i t ]                     7.5195
[ w h a t i s t h a t ]    => [ w h a t ] [ i s t h a t ]                14.9418
[ y o u c a n ]            => [ y o u ] [ c a n ]                         3.3597

Round 2 (The first 12 in 112 decompositions):
[ i s t h a t ]            => [ i s ] [ t h a t ]                         0.6786
[ t h a t ' s ]            => [ t h a t ] [ ' s ]                         3.3377
[ t h a t ' s a ]          => [ t h a t ] [ ' s a ]                       1.1489
[ t h e d o g g i e ]      => [ t h e ] [ d o g g i e ]                  15.4579
[ w h e r e ' s ]          => [ w h e r e ] [ ' s ]                       6.5925
[ i n t h e r e ]          => [ i n ] [ t h e r e ]                       5.4346
[ t h e d o g ]            => [ t h e ] [ d o g ]                         0.2781
[ t h i s i s ]            => [ t h i s ] [ i s ]                         1.6062
[ y o u l i k e ]          => [ y o u ] [ l i k e ]                      10.3563
[ t h e b u n n y ]        => [ t h e ] [ b u n n y ]                    11.6377
[ a n o t h e r o n e ]    => [ a ] [ n o t h e r ] [ o n e ]             2.7744
[ t h e b o o k ]          => [ t h e ] [ b o o k ]                       8.0009

Round 3 (The first 10 in 35 decompositions):
[ w h a t ' s ]            => [ w h a t ] [ ' s ]                         6.4455
[ y o u w a n t ]          => [ y o u ] [ w a n t ]                       8.6686
[ a r e y o u ]            => [ a r e ] [ y o u ]                         3.6702
[ d o y o u ]              => [ d o ] [ y o u ]                           1.4109
[ d o w i t h ]            => [ d o ] [ w i t h ]                         0.8707
[ d i d n ' t ]            => [ d i d ] [ n ' t ]                         0.6807
[ g o o d b y e ]          => [ g o o d ] [ b y e ]                       4.4128
[ w i t h t h e ]          => [ w i t h ] [ t h e ]                       5.6365
[ c o w j u m p i n g o v e r t h e m o o n ] => [ c o w ] [ j u m p ] [ i n g ]
                             [ o v e r ] [ t h e m ] [ o ] [ o n ]       1.2150
[ k n o c k e d t h e m ]  => [ k n o c k ] [ e d ] [ t h e m ]          0.1631
[ p u t i t ]              => [ p u t ] [ i t ]                           0.4277
[ w h a t ' ]              => [ w h a t ] [ ' ]                           0.1275

Round 4 (2 decompositions):
[ d o e s s h e ]          => [ d o e s ] [ s h e ]                       0.0765
[ a n y o u ]              => [ a n ] [ y o u ]                           0.2196
```

Table 3: Sample decompositions of word-clump lexical candidates into finer-grained lexical items during the lexical refinement process

56	7	392	24.7372	[b y e + b y e]
80	5	400	8.9886	[r i g h t]
68	6	408	11.2601	[l i t t l e]
207	2	414	-1.8734	[o k]
71	6	426	10.1195	[n o t h e r]
144	3	432	2.0056	[h i m]
145	3	435	0.4780	[h i s]
109	4	436	4.6727	[h e ’ s]
91	5	455	8.5864	[i t ’ s a]
76	6	456	17.3798	[b l o c k s]
92	5	460	9.5032	[d o n ’ t]
155	3	465	2.5998	[c a n]
162	3	486	-0.1738	[o n e]
122	4	488	5.4852	[i t ’ s]
168	3	504	2.8772	[i n g]
64	8	512	20.7181	[t h a n k y o u]
106	5	530	11.9946	[d a d d y]
268	2	536	-2.9071	[i t]
134	4	536	4.7049	[w i t h]
181	3	543	3.5252	[p u t]
109	5	545	8.1970	[h e l l o]
93	6	558	14.9853	[d o g g i e]
144	4	576	6.0441	[g o o d]
116	5	580	13.5829	[m o m m y]
147	4	588	7.6852	[h a v e]
629	1	629	-5.5904	[a]
160	4	640	6.7353	[y o u r]
161	4	644	6.1727	[o k a y]
65	10	650	41.8614	[p e e k + a + b o o]
372	2	744	-0.2737	[’ s]
194	4	776	7.1221	[l i k e]
275	3	825	2.7419	[a n d]
179	5	895	10.0591	[w a n n a]
224	4	896	7.8500	[b o o k]
329	3	987	0.7991	[s e e]
560	2	1120	-2.3705	[o h]
292	4	1168	4.8850	[h e r e]
342	4	1368	5.3697	[y e a h]
393	4	1572	5.5280	[t h i s]
455	4	1820	7.8423	[l o o k]
421	5	2105	8.9229	[t h e r e]
776	3	2328	1.5757	[t h e]
985	3	2955	5.0111	[y o u]
784	4	3136	7.1216	[w h a t]
856	4	3424	5.5552	[t h a t]

Table 4: Sample finer-grained lexical items output from the lexical refinement process, in ascending order of coverage. Each candiate's count, length, coverage and average DLG are attached on the left

```
 86  6  516  10.4986 [ n o t h e r ]
104  5  520   9.7007 [ w h o ' s ]
110  5  550  12.0695 [ d a d d y ]
110  5  550   8.2140 [ h e l l o ]
139  4  556   5.7148 [ i t ' s ]
113  5  565  12.7322 [ b u n n y ]
145  4  580   4.8412 [ w i t h ]
580  1  580  -5.7134 [ a ]
146  4  584   6.0685 [ g o o d ]
118  5  590  13.6191 [ m o m m y ]
100  6  600  14.7449 [ d r a g o n ]
304  2  608  -1.7303 [ d o ]
308  2  616  -3.4989 [ t o ]
315  2  630  -2.6536 [ i t ]
158  4  632   7.8306 [ h a v e ]
161  4  644   6.1727 [ o k a y ]
 66 10  660  41.9282 [ p e e k + a + b o o ]
135  5  675   6.3450 [ t h o s e ]
236  3  708   0.4372 [ o n e ]
142  5  710  10.3046 [ d o n ' t ]
243  3  729   4.0315 [ p u t ]
385  2  770  -1.5703 [ i s ]
279  3  837   1.4219 [ a r e ]
221  4  884   7.8259 [ b o o k ]
300  3  900   2.8851 [ a n d ]
185  5  925  10.1183 [ w a n n a ]
236  4  944   7.4020 [ y o u r ]
244  4  976   7.5159 [ l i k e ]
244  4  976   5.9158 [ w a n t ]
206  5 1030   9.4411 [ w h e r e ]
175  6 1050  16.2244 [ d o g g i e ]
359  3 1077   4.0248 [ c a n ]
364  3 1092   0.9634 [ s e e ]
221  5 1105  10.8338 [ r i g h t ]
287  4 1148   4.8562 [ h e r e ]
615  2 1230  -2.2253 [ o h ]
346  4 1384   5.3887 [ y e a h ]
471  4 1884   7.9047 [ l o o k ]
456  5 2280   9.0590 [ t h e r e ]
576  4 2304   6.1544 [ t h i s ]
1275 2 2550   1.8949 [ ' s ]
1085 3 3255   2.1182 [ t h e ]
1256 3 3768   5.4795 [ y o u ]
1176 4 4704   6.1291 [ t h a t ]
1274 4 5096   8.0147 [ w h a t ]
```

Table 5: Sample lexical items in the final lexicon after word segmentation using the refined lexicon, in ascending order of coverage. Each candidate's count, length, coverage and average DLG are attached on the left

```
[she's][really][in][to][books][right][now]
[you][want][to][see][the][book]
[oh][look][there]['sa][boy][with][his][hat]
[and][a][doggie]
[oh][you][want][to][look][at][this]
[w'][look][at][this]
[have][a][drink]
[ok][now]
[oh][what]['s][this]
[what]['s][that]
[getit]
[getit]
[getit]
[is][that][for][the][doggie]
[can][you][feed][it][to][the][doggie]
[feed][it]
[oh][put][it][in][ok]
[ok]
[what][are][you][gonna][do]
[i'll][le][ther][play][with][the][se]
[play][with][this][for][aw][hi][le]
```

Figure 7: A fragment of the output from word segmentation using the refined lexicon

namely, the optimal segmentation and lexical refinement) consist of two parts: One is the segmentation result, as illustrated in Figure 7, and the other is the correspondent lexicon, as illustrated in Table 5.

6 Evaluation

In this section we will report the evaluation of our lexical learning algorithms' performance based on the experimental results on the Bernstein corpus. We will first define a number of empirical measures for the evaluation, and then present the learning performance in terms of these measures.

6.1 Evaluation Measures

We are going to use the following empirical measures to evaluate a lexical learner's performance on a given testing corpus.

- Word precision and recall
- Word boundary precision and recall
- Correct character ratio
- Word type precision and recall

The word precision is defined as the proportion of correct words among the learned words, and the word recall as the proportion of real words that are learned by the learner. Given that the input corpus has N words, and the learner learns M words among which C words are correct, the word precision and recall are $\frac{C}{M}$ and $\frac{C}{N}$, respectively, computed in terms of the numbers of word tokens.

However, the measures of word precision and recall do not give any credit to a learning output like `[ithink][itwill][comeout]`, where although none of the chunks in the segmentation is a real word, the learner really detects some regularity within the input data – many word

boundaries are correctly discovered. In order to complement the measures of word precision and recall, we need to have the word boundary precision and recall as evaluation measures.

The word boundary precision is defined as the proportion of correct word boundaries among the word boundaries detected by the learner, and the word boundary recall as the proportion of correct word boundaries that the learner detects. Given that the input corpus has N' word boundaries, and the learner detects M' word boundaries among which C' boundaries are correct, the word boundary precision and recall are $\frac{C'}{M'}$ and $\frac{C'}{N'}$, respectively.

The correct character ratio is the proportion of the entire input corpus in terms of the number of characters that is in correctly learned words. Given an input corpus of L characters long and L' characters out of L are in the correctly learned words, the correct character ratio of the learning is $\frac{L'}{L}$.

The word type precision and recall are the precision and recall in terms of the learned and standard word types in the learned and standard lexicons, respectively, instead of word tokens in the original corpus and the segmentation result by the learning. The word type precision was also called *lexicon precision* in Brent's recent work [2]. These measures form a systematic evaluation for computational lexical learning.

6.1.1 Words versus Morphemes in the Evaluation

However, the above measures would be in vain if it is not clear about what words are in a language. Thus what words are is a critical issue in this evaluation.

The basic rule we opt to follow is that we recognise the orthographic words in the input corpus: spaces are word delimiters. That is, any strings separated by spaces in the input corpus are, basically, recognised as words. This rule is consistent with our principle of minimally necessary change in the data pre-processing: we respect the original corpus as much as we can. Therefore, although we realise that there are many non-conventional word forms in the Bernstein corpus, e.g., "`i gotchyou gotchya gotchya`", "`what d'ya want to do`", "`didja knock'em over`" and "`wa'dja like to call it`", where several words are wrapped up into one "word", we use them as "it is".

Unfortunately, this simple rule does not help solve another problem that is caused by abbreviated (i.e., phonetically reduced) words, e.g., `[-'s]` as in `[that's]`, `[what's]` and `[there's]`, `[-'re]` in `[you're]` and `[there're]`, `[-'ll]`" as in `[i'll]` and `[we'll]`, and `[-n't]` as in `[can't]`, `[don't]`, `[doesn't]` and `[isn't]`. The problem is, if the learner outputs segmentation results like `[that]['s]` and `[does][n't]`, how do we evaluate such cases? Should we count them as correct, or as wrong?

It is difficult to have a clear decision here about correctness. If we *only* count real words in the evaluation, they should certainly be counted as wrong words, because we know for sure that `['s]` and `n't]` are not words. However, our task here is not counting real words; instead, we are to evaluate the performance of an unsupervised lexical learning approach which is expected to simulate human lexical learning mechanisms.

What do infant learners learn in lexical acquisition? Only words? No, they also learn many *morphemes* as lexical items in addition to words. Morphemes are defined in linguistics as the minimal units that have meaning in a language. There are two types of morpheme: one is *free* morpheme, and the other is *bound* morpheme. The free morphemes are words. Most morphemes in a language like English are words. The bound morphemes are the ones that cannot occur

alone by themselves, e.g., [-ing] and [-ed]. The aforementioned reduced word forms such as [-'s] and [-n't] become bound morphemes mostly due to the fact that they lack a vowel to form independent syllables. It is understood that the apostrophe ['] in [-n't] is used to represent a reduced vowel that is not qualified to make a syllable[4]. The number of bound morphemes in a language like English is rather small, and they only can occur in adherence to a word as in [don't]. They are a close class of lexical items in the lexicon of a language.

Now, our problem concerning the question of "what are words?" in the evaluation becomes how do we credit the two types of morphemes that a learner learns in the learning? If we only count the correct words in the learning output, that means that the credit is only given to free morphemes but not to any bound morphemes – an evaluation like this turns out to be imperfect, unfair, and even flawed, in a sense. If we credit a bound morpheme in the same way as a word, it can be quite controversial – those bound morphemes are not words! And we even have no idea about whether a bound morpheme should credited 0.5 or 0.9 as much as a word.

All these unsettled problems indicate that it is inappropriate, and unrealistic, to have a clear-cut means of evaluation for unsupervised lexical learning that has both words and bound morphemes as learning output. The only conceivable way out of this dilemma is that we conduct separate evaluations for the learner's performance on learning words and on learning lexical items including both words and bound morphemes.

To evaluate the performance of learning words, we simply count the chunks in the learning output that match the space-delimited words in the input corpus, and then produce the evaluation results in terms of the above measures. But in the evaluation of the performance of learning words and bound morphemes, the case is still a bit complicated. In general, we take a flexible approach: either the learner recognises, for example, [that's] as one word or as two lexical items [that][’s], we consider both as correct responses.

But this approach is problematic in dealing with the contracted negative marker [-n't], because sometimes the [-] part is not a well-formed word, e.g., [ca][n't] and [wo][n't]. Thus, we need to adjust it accordingly, that is, either the learner recognises [Vn't] as one word a [Vn't] or two lexical items as [V][n't] is considered as correct, conditioned on the fact that the [V] part must form a well-formed word. That is, output chunks like [can't] and [won't] are all considered correct lexical items, but [ca][n't] and [wo][n't] are each recognised as a wrong item and a correct one, instead of as two correct items. In contrast, [do][n't] and [is][n't] are each recognised as two correct items.

The bound morphemes in English texts that we need to consider in the evaluation include, at least, the following: [-'s], [-'d], [-'re], [-'ll], [-'ve] and [-n't], all of which carry an apostrophe. We divide these bound morphemes into two groups. One group, denoted as G1, includes all the above ones but the last. These morphemes only co-occur with a noun. The last one [-n't] forms another group, denoted as G2, which only co-occurs with a verb.

6.2 Learning Performance

The learning performance of the twelve unsupervised lexical learning algorithms, each with a slightly different combination of our OS, LR, WS algorithms and the EM algorithm, are presented in Table 6 and Table 7 with the measures of word and word boundary precision and recall.

[4]Thus, in order to enable the OS + V algorithm to learn bound morphemes such as -n't, we have to tell the learner that ['] is something equivalent to a vowel character.

Learning Algorithms	Word (Token)		Word Boundary		Corr. Char. Ratio (%)
	P (%)	R (%)	P (%)	R (%)	
OS	32.43	31.03	70.48	67.43	33.14
OS+EM	58.66	56.06	83.11	79.42	54.71
OS+LR+WS	61.60	67.99	81.28	89.72	64.10
OS+(LR+WS)x2	58.98	68.49	78.99	91.72	63.17
OS+V	47.99	36.72	85.41	65.36	38.23
OS+V+EM	63.54	49.78	90.71	71.06	49.01
OS+V+LR+WS	74.99	70.92	90.29	85.39	70.04
OS+V+(LR+WS)x2	75.31	73.39	89.63	87.34	71.69
OS+V'	47.42	36.62	84.83	65.50	38.12
OS+V'+EM	63.42	51.12	90.72	73.13	50.51
OS+V'+LR+WS	70.17	70.13	87.58	87.53	68.12
OS+V'+(LR+WS)x2	69.26	72.22	86.06	89.75	68.80

Table 6: Learning performance of the unsupervised lexical learning algorithms on words

Learning Algorithms	Word (Token)		Word Boundary		Corr. Char. Ratio (%)	Counted Morphemes
	P (%)	R (%)	P (%)	R (%)		
	33.86	32.17	71.19	67.65	34.33	+G1
OS	34.04	32.32	71.28	67.68	34.35	+G1+G2
	65.52	60.50	86.57	79.74	59.86	+G1
OS+EM	65.55	60.52	86.58	79.94	59.88	+G1+G2
	69.89	73.77	85.42	90.17	71.33	+G1
OS+LR+WS	70.36	74.09	85.66	90.19	71.75	+G1+G2
	67.08	74.40	83.04	92.10	70.60	+G1
OS+(LR+WS)x2	68.39	75.31	83.69	92.15	71.81	+G1+G2
	49.33	37.81	85.78	65.76	39.28	+G1
OS+V'	49.55	37.95	85.89	65.78	39.41	+G1+G2
	69.45	54.66	93.68	73.73	54.38	+G1
OS+V'+EM	69.95	54.94	93.94	73.78	54.70	+G1+G2
	79.42	75.87	92.20	88.08	75.42	+G1
OS+V'+LR+WS	79.92	76.17	92.45	88.11	75.82	+G1+G2
	78.76	78.25	90.81	90.23	76.62	+G1
OS+V'+(LR+WS)x2	80.11	79.07	91.49	90.30	77.74	+G1+G2

Bound morphemes in group G1:	[-'s], [-'d], [-'re], [-'ll], [-'ve]
Bound morphemes in group G2:	[-n't]

Table 7: Learning performance of the unsupervised lexical learning algorithms on word and bound morphemes

Table 6 presents their performance in learning real words, and Table 7 their performance in learning words and bound morphemes.

The performance of the OS and OS+V' algorithms incorporated in the EM algorithm is presented in Figure 8. We can see that each of the two programs converges very quickly towards the top of their performance. In general, both algorithms' performance drops significantly in the second iteration, except for OS+EM's word precision and word boundary precision, which continue going up. Then, all measures increase rapidly in the next five iterations. After that, the growth slows down significantly in the next ten iterations, and all measures reach their local maxima gradually.

We have observed that the OS and OS+EM algorithms have a better balance between precision and recall than the OS+V' and OS+V'+EM algorithms, as shown in the middle table in Table 8. The former two algorithms' precision and recall for word and word boundary show a difference, defined as $|P - R|$, less than 3.69 percentage points and a divergence rate, defined as $\frac{|P-R|}{\min(P,R)}$, less than 4.65%; whereas the latter two algorithms have a difference of precision and recall in the range of 10 to 20 percentage points and a divergence rate in the range of 20% to 30%. But the EM algorithm seems not to enlarge this difference and divergence rate: OS+EM has a balance of precision and recall as good as OS, and OS+V'+EM has a slightly better balance of precision and recall than OS+V'.

We can see the effect of the EM algorithm on improving the learning performance of the OS and OS+V' algorithms. As shown in the middle table in Table 8, the EM algorithm increases the OS algorithm's word precision and recall both by about 81% and its word boundary precision and recall both by about 18%. In contrast, the EM algorithm appears less effective, but still quite effective, on the OS+V' algorithm, in that it increases the word precision and recall by about 34% and 40%, respectively, and increases the word boundary precision and recall by about 7% and 12%, respectively. This loss in effectiveness is, most likely, due to the effect of the vowel constraint.

The bottom table in Table 8 presents the effect of the vowel constraint on learning performance, with V' (a looser constraint) as an example. It shows that except for the case of working with the EM algorithm, the vowel constraint consistently enhances the learning performance by improving the word precision and recall and word boundary precision, at the price of lowering the word boundary recall slightly. While working with the EM algorithm, the vowel constraint improves both the word precision and word boundary precision, by about 8% and 9%, respectively, at the price of lowering the correspondent recalls by about -9% and -8%, respectively, and also lowering the correct character ratio by 7.68%. These results indicate that the vowel constraint is not a good co-operator with the EM algorithm.

The EM algorithm can only reach a local minimum; and human learners do not learn in the same way by going through the input data many times. We are more interested in pursuing learning algorithms that can simulate human lexical learning better than the EM algorithm, in terms of both its learning strategies and learning performance. We have implemented several such learning algorithms, including the OS+LR+WS, OS+V+LR+WS and OS+V'+LR+WS algorithms. Their performance in learning words and in learning words and bound morphemes has been shown in Table 6 and 7, respectively, and the comparison of their performance and their correspondent EM algorithms' performance is presented in Table 9. We can see that the unsupervised lexical learning algorithms have a much better performance than the EM algorithm: the word precision is better by 5-18%, the word recall is better by 21-42%, the word boundary precision

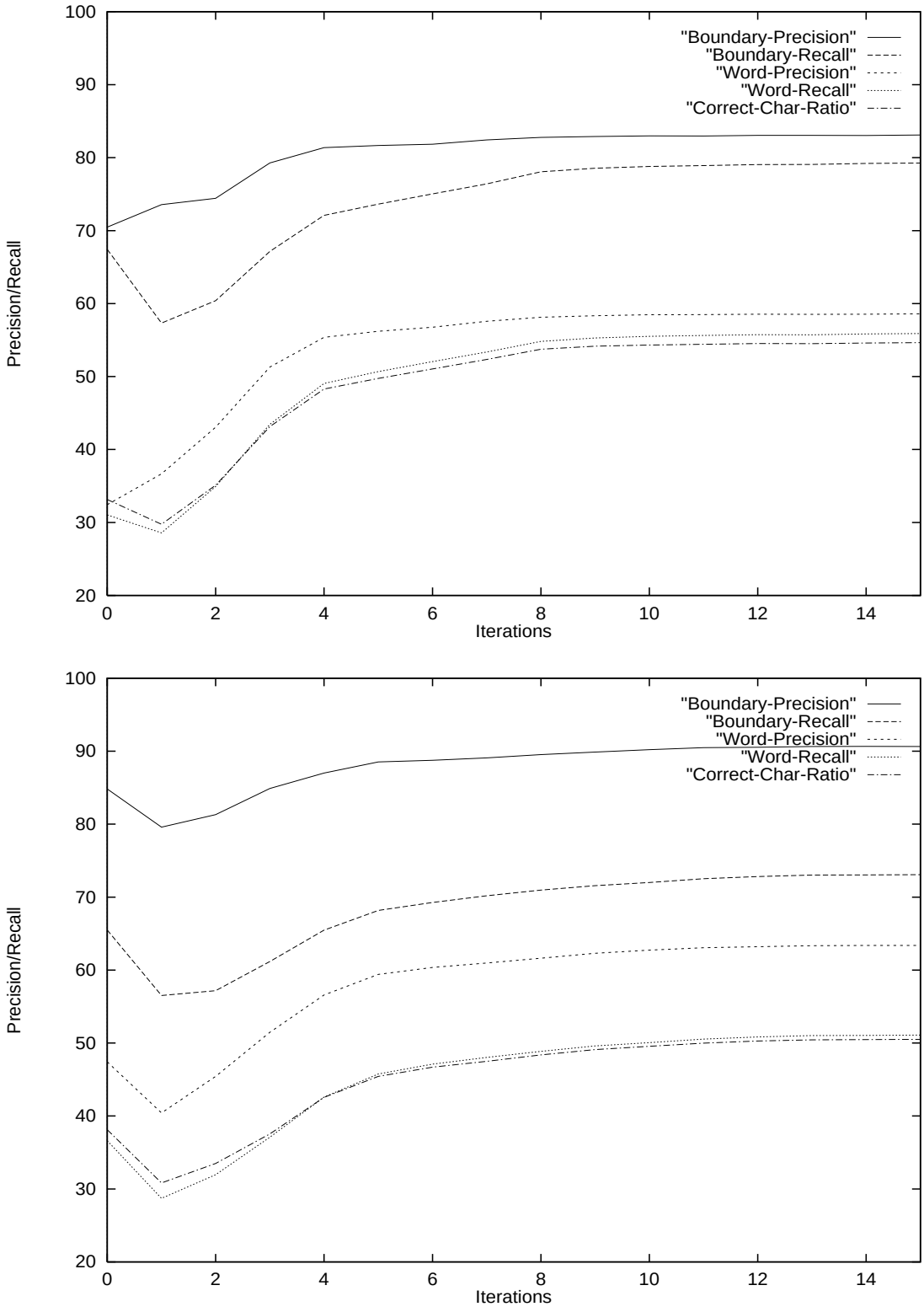

Figure 8: Performance of the `OS` and `OS+V'` algorithms in EM iterations

Algorithms	Word				Word Boundary			
	P (%)	R (%)	|P − R|	D (%)	P (%)	R (%)	|P − R|	D (%)
OS	32.43	31.03	1.40	4.51	70.48	67.43	3.05	4.52
OS+EM	58.66	56.06	2.60	4.64	83.11	79.42	3.69	4.65
OS+V'	47.42	36.62	10.80	29.49	84.83	65.50	19.33	29.51
OS+V'+EM	63.42	51.12	12.30	20.06	90.72	73.13	17.59	24.05

Algorithms	OS+EM				OS+V'+EM			
	Word		Word Boundary		Word		Word Boundary	
	P (%)	R (%)	P (%)	R (%)	P (%)	R (%)	P (%)	R (%)
Beginning	32.43	31.03	70.48	67.43	47.42	36.62	84.83	65.50
End	58.66	56.06	83.11	79.42	63.42	51.12	90.72	73.13
Increment	26.23	25.03	12.63	11.99	16.00	14.50	5.89	7.63
Incr. Rate	80.88	80.66	17.76	17.78	33.74	39.60	6.94	11.65

Learning Algorithms	Word (Token)		Word Boundary		Corr. Char. Ratio (%)
	P (%)	R (%)	P (%)	R (%)	
OS	32.43	31.03	70.48	67.43	33.14
OS+V'	47.42	36.62	84.83	65.50	38.12
Increment	14.99	5.59	14.35	-1.92	4.98
Incr. Rate (%)	46.22	18.01	20.36	-2.86	15.03
OS +EM	58.66	56.06	83.11	79.42	54.71
OS+V'+EM	63.42	51.12	90.72	73.13	50.51
Increment	4.76	-4.94	7.61	-6.29	-4.20
Incr. Rate (%)	8.11	-8.81	9.16	-7.92	-7.68
OS +LR+WS	61.60	67.99	81.28	89.72	64.10
OS+V'+LR+WS	70.17	70.13	87.58	87.53	68.12
Increment	9.00	2.14	6.30	-2.19	4.02
Incr. Rate (%)	14.61	3.15	7.75	-2.44	6.27
OS +(LR+WS)x2	58.98	68.49	78.99	91.72	63.17
OS+V'+(LR+WS)x2	69.26	72.22	86.06	89.75	68.80
Increment	10.28	3.73	7.07	-1.97	5.63
Incr. Rate (%)	17.43	5.45	8.95	-2.15	8.91

Table 8: The effectiveness of the EM algorithm and the vowel constraint

Learning Algorithms	Word (Token)		Word Boundary		Corr. Char. Ratio (%)
	P (%)	R (%)	P (%)	R (%)	
OS+EM	58.66	56.06	83.11	79.42	54.71
OS+LR+WS	61.60	67.99	81.28	89.72	64.10
Increment	2.94	11.93	-1.83	10.30	9.39
Incr. Rate (%)	5.01	21.28	-2.20	12.97	17.16
OS+V+EM	63.54	49.78	90.71	71.06	49.01
OS+V+LR+WS	74.99	70.92	90.29	85.39	70.04
Increment	11.54	21.14	-0.42	14.33	21.03
Incr. Rate (%)	18.02	42.47	-0.46	20.17	42.91
OS+V'+EM	63.42	51.12	90.72	73.13	50.51
OS+V'+LR+WS	70.17	70.13	87.58	87.53	68.12
Increment	6.75	19.01	-3.14	14.40	17.61
Incr. Rate (%)	10.64	37.19	-3.46	19.69	34.86

Learning Algorithms	Word (Token)		Word Boundary		Corr. Char. Ratio (%)
	P (%)	R (%)	P (%)	R (%)	
OS+EM	65.55	60.52	86.58	79.94	59.88
OS+LR+WS	70.36	74.09	85.66	90.19	71.75
Increment	4.81	13.57	-0.92	10.25	11.87
Incr. Rate (%)	7.34	22.42	-1.06	12.82	19.82
OS+V'+EM	69.95	54.94	93.94	73.78	54.70
OS+V'+LR+WS	79.92	76.17	92.45	88.11	75.82
Increment	9.97	21.23	-1.49	14.33	21.12
Incr. Rate (%)	14.25	38.64	-1.59	19.42	38.61

Table 9: Comparison of learning performance on words and on words and bound morphemes: unsupervised lexical learning algorithms versus the EM algorithm

is slightly lower, at most by 3.5%, the word boundary recall is better by 13-20%, and the correct character ratio is better by 17-42%. The superiority of unsupervised lexical learning through optimal segmentation, lexical refinement and word segmentation is clearly overwhelming.

The effect of repeating the `(LR+WS)` part in the unsupervised lexical learning appears insignificant. It increases the recall slightly but decreases, sometimes, the precision to a similar scale. For example, in the experiment of learning words and bound morphemes, the `OS+V+(LR+WS)x2` algorithm, in comparison with the `OS+V+LR+WS` algorithm, increases the word recall by 1.2 percentage points at the price of lowering the word precision by 2 points and increases the word boundary recall by about 2 points at the price of lowering the word boundary precision by about 2 points. The `OS+V'+(LR+WS)x2` algorithm demonstrates the best gain by the `(LR+WS)x2` part on learning words and morphemes: a gain of 2 percentage points in word recall with no loss in word precision, a gain of 2.2 points in word boundary recall at the price of 1 point loss in word boundary precision, and a gain of about 2 points in correct character ratio.

When bound morphemes are counted as correctly learned lexical items in the learning output, the learning performance of the `OS+LR+WS` and `OS+V'+LR+WS` algorithms goes up significantly. These two algorithms' precision and recall on lexical items, precision and recall on

Learning Algorithms	Lexical Type	Lexical Item		Lexical Boundary		Corr. Char. Ratio (%)
		P (%)	R (%)	P (%)	R (%)	
OS+LR+WS	Word	61.60	67.99	81.28	89.72	64.10
	Word + Morph.	70.36	74.09	85.66	90.19	71.75
	Increment	8.76	6.10	4.38	0.47	7.65
	Incr. Rate (%)	14.22	8.97	5.34	0.52	11.93
OS+V'+LR+WS	Word	70.17	70.13	87.58	87.53	68.12
	Word + Morph.	79.92	76.17	92.45	88.11	75.82
	Increment	9.75	6.04	4.87	0.58	7.70
	Incr. Rate (%)	13.89	8.61	5.56	0.66	11.30

Table 10: Comparison of learning performance on words and on words and morphemes

Learning Algorithms	Lexical Type	Lexical Item		Difference from MBDP-1	
		P (%)	R (%)	P (%)	R (%)
OS+LR+WS	Word	61.60	67.99	−9.40 (−13.2%)	−3.01 (−4.2%)
	Word + Morph.	70.36	74.09	−0.64 (−0.9%)	+2.09 (+2.9%)
OS+V+LR+WS	Word	74.99	70.92	+3.99 (+5.6%)	−1.08 (−1.5%)
OS+V'+LR+WS	Word	70.17	70.13	−0.93 (−1.3%)	−0.97 (−1.3%)
	Word + Morph.	79.92	76.17	+8.08 (+11.4%)	+4.17 (+5.8%)

Learning Algorithms	Lexical Type	Lexical Item			F Difference from MBDP-1
		P (%)	R (%)	F (%)	
OS+LR+WS	Word	61.60	67.99	60.86	−10.64 (−14.9%)
	Word + Morph.	70.36	74.09	72.18	+0.68 (+0.1%)
OS+V+LR+WS	Word	74.99	70.92	72.90	+1.40 (+2.0%)
OS+V'+LR+WS	Word	70.17	70.13	70.15	−1.35 (−1.9%)
	Word + Morph.	79.92	76.17	78.00	+6.50 (+9.1%)

Table 11: Comparison of our lexical learning algorithms' performance with the state-of-the-art performance

lexical boundaries, and correct character rate are, respectively, about 13%, 9%, 5.5%, 0.5% and 11-12% higher than those on words. Of all these increments, only the increment of the boundary recall is not notably significant.

In general, the learning performance of our algorithms compares favourably with the state-of-the-art performance of Brent's MBDP-1 algorithm. The MBDP-1 algorithm is estimated to have an average word precision of around 71% and word recall of around 72%. The upper table in Table 11 shows the comparison in terms of the difference in word precision and recall.

In order to make the comparison a bit clearer, we can use the F measure to combine the precision and recall together for the overall learning performance. The F measure is a variant of van Rijsbergen's E measure introduced in [25]: $F = 1 - E$. The F measure is defined as

$$F = \frac{1}{\alpha\frac{1}{P} + (1-\alpha)\frac{1}{R}} \tag{10}$$

In general, we consider precision and recall equally important, and consequently choose a value

of $\alpha = 0.5$. Accordingly, we have

$$F = \frac{2PR}{P+R} \tag{11}$$

Following this formula, the average overall learning performance of the MBDP-1 algorithm is $\frac{2*71*72}{71+72} = 71.5$, as we can estimated. The bottom table in Table 11 shows the comparison of our algorithms' overall performance with that of MBDP-1.

The result of this comparison indicates that our best algorithm for word learning, namely, the `OS+V+LR+WS` algorithm, has an overall performance of learning words that compares favourably with MBDP-1's overall performance. If bound morphemes are considered as correctly learned lexical items, the `OS+LR+WS` algorithm has a learning performance as good as MBDP-1, whereas the `OS+V'+LR+WS` algorithm has a significantly better performance[5].

If Brent had evaluated MBDP-1's learning performance with the word boundary precision and recall and with the correct character ratio, we could have a more thorough comparison between our learning algorithms and MBDP-1. Based on the above comparison, we may not be certain that our learning approach really outperforms the MBDP-1 algorithm, because many factors in the learning algorithms and related to the testing data preparation are different. For example, MBDP-1 is an incremental online learning algorithm, whereas ours are not; onomatopoeia and interjections are cleaned out from the testing data for MBDP-1 but kept in the testing data for our algorithms; MBDP-1 learns from phonetic transcripts and our algorithms learn from orthographic transcripts. All these factors mean that the above comparison carries a certain degree of roughness.

However, the comparison has no doubt provided adequate evidence for the conclusion that our learning approach reaches the level of the state of the art of unsupervised lexical learning.

7 Discussion

Although our lexical learning approach has turned out to show outstanding performance in learning lexical items, including words and bound morphemes, we also have observed that there is still quite some room for further improvement. In this section, we will analyse a number of problems that our unsupervised lexical learners encountered in the experiments and discuss possible solutions.

7.1 Negative DLG Segmentation

The first problem that still needs to be resolved is the negative DLG segmentation problem: some low frequency words in an utterance, which may or may not exist in the refined lexicon, can cause the segmentation of the entire utterance to have a segmentation with a negative DLG. Table 12 lists many examples of negative DLG segmentations output from the `OS+V'LR+WS` algorithm, with the frequency of the problem-causing words given at the right. Actually, it is not a problem for the learner not to be able to recognise the low frequency words. It is absolutely normal for all unsupervised lexical learning algorithms based on co-occurring statistics not to learn most of these "bad" words. What is really a problem in our DLG-based learning algorithm is that when a "bad" word is lost in this way, it looks as though this bad word interferes with

[5]Notice that `OS+V`-based algorithms, including `OS+V+LR+WS`, do not learn any bound morphemes like those listed in G1 and G2, which have an apostrophe for the reduced vowel.

	DLG	Examples of segmentation	Freq.
1	-35.1325	asmileithasasmile	2
2	-32.6349	i just realizedhowitworks	1
3	-14.0336	that 'sawickedlaugh you monkey	1
4	-29.7511	one that willflapinthebreeze	1
5	-35.1325	iwanttoshowyousomethin'	3
6	-22.5597	does the chair partflattenitout	1
7	-34.0216	okletsgetdownandgettothese toys	2
8	-35.1232	ahaahaahaahawhatelsedoeshe say	6
9	-27.7897	i don't know whyshehatesitsomuch	2
10	-35.1325	ohigotigotanidea	1
11	-5.3626	you 'reintodestruction aren't you Alice	1
12	-29.5773	that 'swhywehadyourestrained	1
13	-22.5104	you can spankitifitbites you	2
14	-22.7013	and this boy 's put tingonhisshirt	3
15	-26.2659	could n'tpossiblybebecauseyourmother'shadaphoneattached	1, 2
16	-15.7839	yeah that 's what the horsedoesyeahp	1
17	-0.7694	wanna look at look at look this onehaspaperpages you might	2, 1
18	-24.4458	that sanowlohtheblocksfell over	3
19	-33.1854	oh and the kidsaresayin'byeseethey'rewavin'	3, 1
20	-26.0025	let's tryawhthisisa snap	(noise)
21	-12.6852	you 'renotatallinterestedin this dragon ' think	2
22	-35.1325	igotchyougotchyagotchya	2, 2
23	-18.1525	do you re member what wesaidthelast time	4
24	-25.2219	mightbehavingtrouble onhisknees	1, 5
25	-22.7947	look sto me like you 'remakinghimdance	1
26	-22.0512	but he's gonna triponhisshoelacestiesomebows	1
27	-28.0104	like adolphinit'sawhale	1
28	-35.1325	hehassomuchhairyoucan'tseehisneck	6
29	-35.1325	yastrapityagoanditsticks	4, 2
30	-29.7628	yeah Iwishitwereaburry	1
31	-13.5204	you know what it's supposedtobe	2
32	-21.7612	it's supposedtobea pieceof steak	2
33	-31.6725	growlswhotookmy steak	1
34	-16.0308	maybe the drayon'dliketobe inthe high chair	1
35	-7.4811	ithink hemightbetoosm all for this high chair	1
36	-20.2497	you know Michael'sgonnagoonvacation today	2, 1
37	-19.2983	it's notakiteitthat'sthesteamcom ing outof the food	4, 2

Table 12: Examples of segmentations with negative DLG

Learning output	Original segmentation
ohigotigotanidea	oh i got i got an idea#
i got blocks athome	i got blocks at home#
oh i got them	oh i got them#
a hi got your nose	ah i got your nose#
i got ya finger	i got ya finger#
wait now i got ta fix it	wait now i gotta fix it#
i got it	i got it#
i got you	i got you#
i got you	i got you#
i got your hand	i got your hand#
i got you	i got you#
i got you lemme open thedoor	i got you lemme open the door#
it's got an a@lb@lc@l	it's got an a@l b@l c@l#

Figure 9: Word clumps due to negative DLG *vs.* correct segmentation

the recognition of other words. For example, in Table 12, `[lets]` in line 7 grabs seven words, namely, `[ok]` and `[get down and get to these]`, into the same clump; `[idea]` in line 10 grabs six other words `[oh i got i got an]` – whereas in almost all other utterances these six words are properly recognised by the unsupervised learner, as exemplified by the output from the `OS+V'+LR+WS` algorithm as below, in comparison with all utterances involving `[i got]` and `[got an]` in the original input corpus in the right column[6]:

Therefore, the key to solving this word-clumping problem is to find a principled way to protect the other words from being corrupted by erroneous recognition of a bad word. We say "principled way" because a cognitively sound strategy of word segmentation when there are unknown words. In our learning approach we assume the same DLG optimisation based approach to word segmentation as to lexical learning.

An ideal strategy is one that can incorporate the advantages of a DLG-based approach but avoid its disadvantages in dealing with the negative DLG problem. It is highly possible that word segmentation by human subjects with an existing lexicon is an optimisation process with a different goodness measure than the one for their lexical learning. It can be a strategy as simple as maximal match segmentation (MMS): scanning through the input, outputting the longest matched word and then moving on to the next word, and continuing to work this way until the entire input utterance is finished. It is also possible that the MMS strategy is incorporated into the DLG optimisation based segmentation. One possible approach to the incorporation would be, whenever a negative DLG segmentation is encountered, to apply the MMS to save as many good words as possible from the clumping effect with "bad" word(s). According to our statistics, 5.76% of the learning output is in such word clumps. That means, if we could have a strategy that can perform word segmentation on this portion as well as on the rest, our learning algorithms would enhance their learning performance by $5.76\% * \frac{70\%}{1-5.76\%} = 4.28\%$. This would

[6]In the original orthographic text of Bernstein corpus, `a@l`, `b@l` and `c@l` denote single letters `A`, `B` and `C`, respectively

Learning Algorithms	Learned Word Types	Correct Word Types	Word Type	
			Precision (%)	Recall (%)
OS	1,922	394	20.50	22.25
OS+ LR+WS	1,046	400	38.24	22.59
OS+(LR+WS)x2	909	390	42.90	22.02
OS+V	3,036	601	19.64	33.94
OS+V+ LR+WS	1,914	568	29.68	32.07
OS+V+(LR+WS)x2	1,792	557	31.08	31.43
OS+V'	2,916	604	20.71	34.11
OS+V'+ LR+WS	1,582	565	35.71	31.90
OS+V'+(LR+WS)x2	1,492	558	37.42	31.49

Table 13: Word type precision and recall of the unsupervised lexical algorithms

be a very significant improvement.

The problem of negative DLG segmentation deserves much research effort in our future work.

7.2 Word Type Precision and Recall

In contrast to the word token precision and recall, which are usually at the level of 70%, the word type precision and recall of our learning algorithms appear low, at the level of slightly higher than 30%, as listed in Table 13. The number of word types in the original input corpus is 1771.

As a matter of fact, the word type precision and recall in unsupervised lexical learning are commonly at this level, or even lower. There has been no report on them, except for the word type precision of the MBDP-1 algorithm reported in [2]. MBDP-1's word type precision starts at about 36% and grows to 54% in its incremental learning on the Bernstein corpus. All other algorithms reported in [2] have a word type precision beneath 30% on average.

From Table 13 we can see that the OS+V' based algorithms have the best performance, and that when the LR+WS is applied one more time, the precision increases significantly and the corresponding recall decreases slightly. Both our OS+V and OS+V' based algorithms learn about 1/3 of the word types in the input corpus.

How can such a low word type recall enable the word token precision and recall, as we reported above, at the level of 70%? The answer is given in Figure 10: the top 500 word types, either in the frequency or coverage ranking, cover about 90% of the input corpus. Our learning algorithms, e.g., OS+V+LR+WS and OS+V'+LR+WS, learn about 550 words correctly, most of which are frequent words. It is not surprising that these words cover more than 70% of the input corpus.

Like other statistically based learning algorithms, our algorithms make fewer errors in learning high frequency words than in learning low frequency words. The word type precision and recall of our lexical learning versus word frequency rank are presented in the two figures in Figure 11. The first figure is plotted in terms of the word frequency in the learning output and the lower in terms of the frequency in the input corpus. The diamond-dots plot the precision or recall at each frequency rank, and the solid lines plot the average precision or recall over word

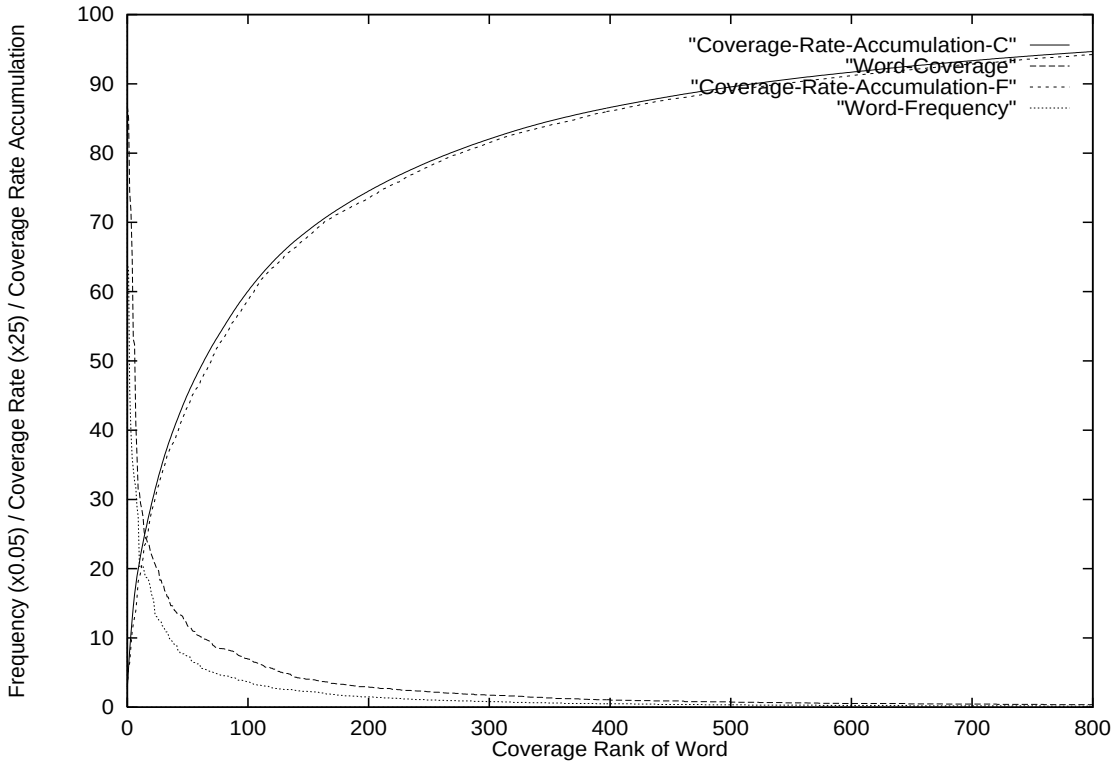

Figure 10: Word coverage rate versus frequency and coverage rank of word

types up to a frequency rank. We can see from the upper figure that the word type precision over frequent words is very high: the average precision up to the first 100 ranks (out of 147) is above 80%, and the learned words are not 100% correct only in 12 ranks out of the first half, roughly 75, of all ranks. We can also see from the lower figure that the word type recall over frequent words is also very high: the average recall up to the first 100 ranks (out of 147) is above 80%, and there are only 16 ranks in the first half, roughly 75, of all ranks in which the words are not 100% correctly learned. The low overall word type precision and recall is determined by the fact that the word type number increases dramatically in the last 10 frequency ranks, where the precision and recall both drop very rapidly.

So, the focus of enhancing the word type precision and recall is on the enhancement of the precision and recall of learning low frequency words.

7.3 Other Problems

In addition to the negative DLG segmentation problem and the problem of low word type precision and recall, there are also other problems that hinder our learning algorithms from performing or scoring any better. Some of these problems are inherent in the input data, e.g., the inconsistency and data noise in the transcripts. Some are related to our evaluation criteria, e.g., `[-ing]` and `[-ed]` are not counted as creditable morphemes in the learning output, because they are another type of morpheme categorically different from abbreviated forms of existing words.

Some problems are directly related to the behaviour of the DLG-based leaner, e.g., in the Bernstein corpus, the word `[balloon]`, with 46 occurrences (a very high frequency), is correctly

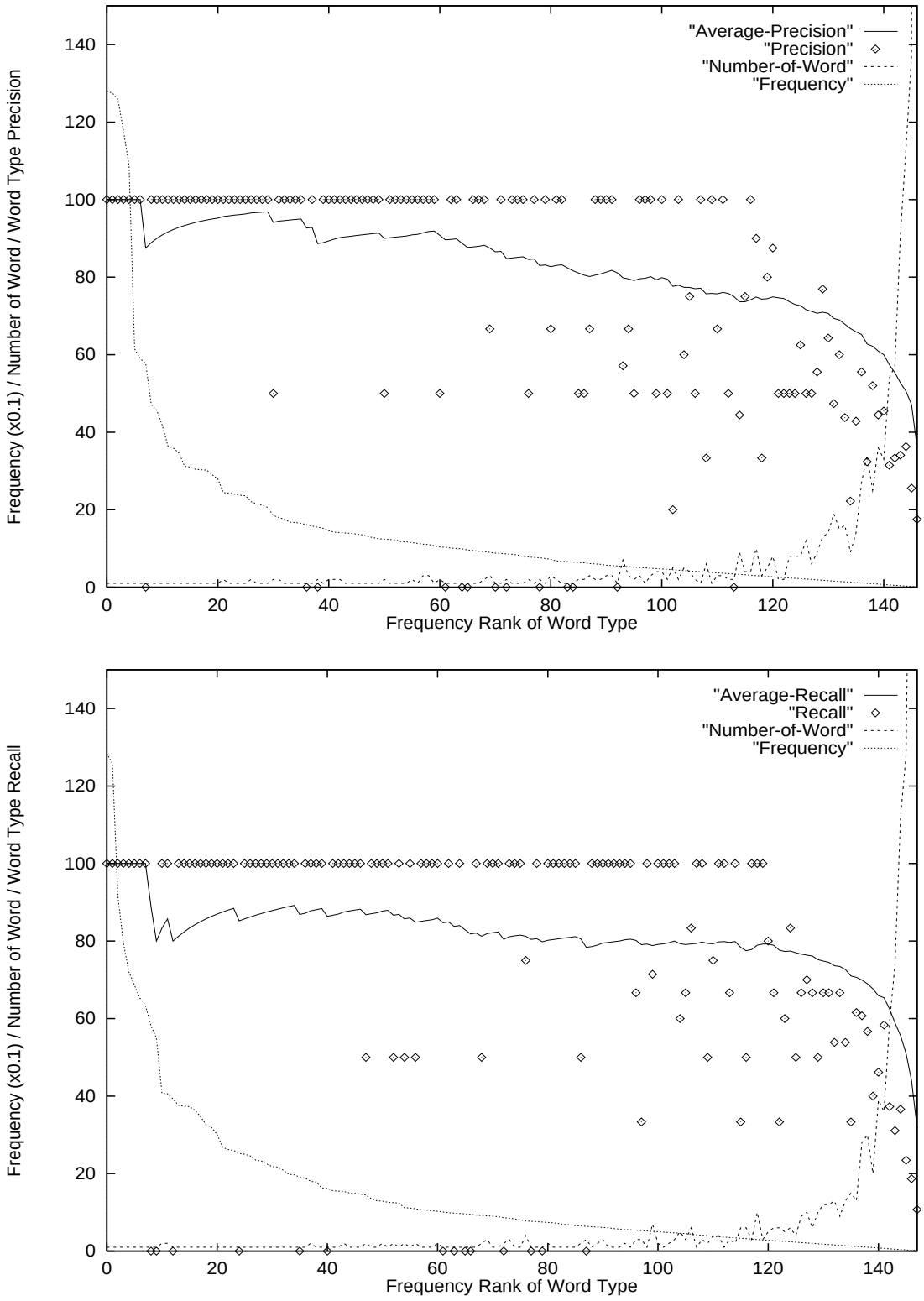

Figure 11: Word type precision and recall in terms of frequency rank of word

recognised 42 times but erroneously divided into [ball][o][on] 3 times, and a more frequent word [another], with 86 occurrences, is always divided into [a][nother][7]. Many frequent words or noun compounds, e.g., [instead] and [golfball], in some other child-directed corpora, are also given abnormal segmentations by the DLG-based learning algorithms. The word [instead], with 25 occurrences, is always segmented into [i][nstead], and the compound [golfball], with 19 occurrences, is divided into [golf][ball] 8 times and recognised as a single word 11 times.

We still do not quite understand such unusual behaviour in a DLG-based lexical learner, because according to the DLG optimisation, the learner should choose [another] and [instead] instead of [a][nother] and [i][nstead], because [another] and [instead] have the same frequency as [nother] and [nstead], respectively, but each has a greater length and therefore a greater (positive) DLG. A lexical learner based on the DLG optimisation should select these longer words instead of the shorter ones. Why does it learn these unexpected words?

These are interesting problems that deserve more research effort. We have a reasonable assumption, namely, the least effort principle, as the starting point for our study. And we have developed an elegant computational theory for the unsupervised lexical learning based on the MDL principle and, accordingly, formulated the DLG goodness measure for selecting word candidates. We also have implemented a number of sound learning programs based on DLG optimisation that can learn most words correctly from the input corpus. However, a DLG lexical learner still has some unexpected behaviours beyond our understanding for the time being. We need to have a thorough understanding of them in order to advance the computational studies on human cognitive mechanisms for lexical learning based on our current achievements with the LDG optimisation approach.

8 Conclusions

We have presented a novel approach of unsupervised lexical learning via compression, including its assumptions, underlying theories, a goodness measure for computing the compression effect of extracting word candidates, an optimal segmentation algorithm following this goodness measure to word candidates, and a learning model to apply this algorithm to derive finer-grained lexical items. Experiments on large-scale corpus of child-directed speech show that its performance compares favourably to the state-of-the-art performance of unsupervised lexical learning. This performance indicates the validity and the effectiveness of the learning approach and the appropriateness of the implementation.

The unsupervised lexical learning is realised by an algorithm to achieve the DLG optimisation over input utterances following the MDL principle. The representation formalism for the learning is trivially simple: each lexical item is represented as a string, with one parameter, namely, its frequency – each lexical item's LDG is calculated in terms of its frequency. The Viterbi algorithm is exploited to search for the segmentation of an utterance that gives the greatest sum of DLG over its segments.

The lexical learning process in our computational approach consists of three phases (or learning modules), namely, DLG-based optimal segmentation of input utterances into lexical candidates, lexical refinement to divide the word-clump candidates into individual words, and then word segmentation in terms of lexical items acquired in the previous two phases. This

[7] Native speakers actually say "that's a whole nother thing".

lexical learning model is consistent with human infants' behaviours in lexical learning: they recognise many word clumps as individual lexical items and later divide them into individual words when they are exposed to more language evidence supporting the decomposability of the clumps [19]. In our approach each of the three phases involves an application of the same optimal segmentation algorithm for DLG optimisation with a different set of word candidates, or, a different word space.

We have developed twelve unsupervised lexical learning algorithms each with a different combination of learning modules, parameters and constraints, and also developed a comprehensive evaluation approach based on seven evaluation measures to systematically examine their learning performance on the orthographic texts of the Bernstein corpus of child-directed speech. The evaluation measures are word precision and recall, word boundary precision and recall, word type precision and recall, and correct character ratio. This is the most comprehensive evaluation approach ever applied in the field of computational lexical learning.

The top performance of our DLG-based unsupervised learning of words and of words and bound morphemes is achieved, respectively, by two typical unsupervised lexical learning algorithms involving the three phases, namely, the `OS+V+LR+WS` and `OS+V'+LR+WS`. The best performance in learning words is 75% precision, 71% recall and, accordingly, $F = 73\%$. This performance compares favourably with the state-of-the-art performance achieved by Brent's MBDP-1 algorithm on the same child-directed speech corpus. Our best performance in learning words and bound morphemes is 80% precision, 76% recall and $F = 78\%$ – this F score is 5 percentage points higher, by an increment of 6.85%.

In addition to the comprehensive evaluation, we have also analysed a number of problems that the DLG-based lexical learning approach encounters, including the negative DLG segmentation problem and the low precision and recall on word type. The analysis points to a direction for future work.

Acknowledgements

The author wishes to thank Yorick Wilks for his enthusiastic support, advice and various kinds of help that have enabled this study, and thank Mign Li and Paul Vitányi for helpful discussions on Kolmogorov complexity, learning via compression, MDL and other theoretical issues related to this work. Sincere thanks also go to Randy LaPolla and Lisa Raphals for their helps, valuable comments and advice that have improved this paper significantly. The author is responsible for all remaining errors.

References

[1] N. Bernstein-Ratner. The phonology in parent child speech. In K. Nelson and A. van Kleeck, editors, *Children's Language*, volume 6. Erlbaum, Hillsdale, NJ, 1987.

[2] M. R. Brent. An efficient, probabilistically sound algorithm for segmentation and word discovery. *Machine Learning*, 34:71–106, 1999.

[3] M. R. Brent and T. A. Cartwright. Distributional regularity and phonological constraints are useful for segmentation. *Cognition*, 61:93–125, 1996.

[4] G. Chaitin. On the length of programs for computing finite binary sequences. *J. Assoc. Comput. Math.*, 13:547–569, 1966.

[5] N. Chomsky. *Syntactic Structure*. Mouton, Hague, 1957.

[6] N. Chomsky. *The Minimalist Program*. MIT Press, Cambridge, MA., 1995.

[7] T. M. Cover and J. A. Thomas. *Elements of Information Theory*. John Wiley and Sons, Inc., New York, 1991.

[8] C. de Marcken. The unsupervised acquisition of a lexicon from continuous speech. Technical Report A.I. Memo No. 1558, AI Lab., MIT, Cambridge, Massachusetts, November 1995.

[9] C. de Marcken. *Unsupervised Language Acquisition*. PhD thesis, MIT, Cambridge, Massachusetts, 1996.

[10] C. Kit. *Unsupervised Lexical Learning as Inductive Inference*. PhD thesis, University of Sheffield, UK, 2000.

[11] C. Kit and Y. Wilks. The Virtual Corpus approach to deriving n-gram statistics from large scale corpora. In C. Huang, editor, *Proceedings of 1998 International Conference on Chinese Information Processing*, pages 223–229, Beijing, November 1998.

[12] C. Kit and Y. Wilks. Unsupervised learning of word boundary with description length gain. In M. Osborne and E. T. K. Sang, editors, *CoNLL-99*, pages 1–6, Bergen, June 1999.

[13] A. N. Kolmogorov. Three approaches for defining the concept of "information quantity". *Problem of Information Transmission*, 1:4–7, 1965.

[14] M. Li and P. M. B. Vitányi. *Introduction to Kolmogorov Complexity and its Applications*. Springer-Verlag, New York, 1993. Second edition, 1997.

[15] B. MacWhinney. *The CHILDES Database*. Discovery Systems, Dublin, OH, 1991.

[16] B. MacWhinney and C. Snow. The child language data exchange system. *Journal of Child Language*, 12:171–296, 1985.

[17] U. Manber and E. Myers. Suffix array: a new method for on-line string searches. In *First ASM-SIAM Symposium on Discrete Algorithms*, pages 319–327, Providence, 1990. American Mathematical Society.

[18] D. C. Olivier. *Stochastic Grammars and Language Acquisition Mechanisms*. PhD thesis, Harvard University, Cambridge, MA, 1968.

[19] A. Peters. *The Units of Language Acquisition*. Cambridge University Press, Cambridge, England, 1983.

[20] J. Rissanen. Modelling by shortest data description. *Automatica*, 14:465–471, 1978.

[21] J. Rissanen. *Stochastic Complexity in Statistical Inquiry*. World Scientific, N.J., 1989.

[22] J. Rissanen and E. S. Ristad. Language acquisition in the MDL framework. In E. Ristad, editor, *Language Computations*. American Mathematical Society, Philadelphia, PA, 1994.

[23] C. Shannon. A mathematical theory of communication. *Bell System Technical Journal*, 27:379–423, 623–656, 1948.

[24] R. J. Solomonoff. A formal theory of inductive inference, part 1 and 2. *Information Control*, 7:1–22, 224–256, 1964.

[25] C. J. van Rijsbergen. *Information Retrieval*. Butterworths, London, 2nd edition, 1979.

[26] A. Venkataraman. A statistical model for word discovery in transcribed speech. *Computational Linguistics*, 27(3):352–372, 2001.

[27] P. M. B. Vitányi and M. Li. On prediction by data compression. In *Proc. 9th European Conference on Machine Learning*, pages 14–30, Heidelberg, 1997. Springer-Verlag. Lecture Notes in Artificial Intelligence, Vol. 1224.

[28] P. M. B. Vitányi and M. Li. Minimum description length induction, Bayesianism, and Kolmogorov complexity. *IEEE Transactions On Information Theory*, IT-46(2):446–464, 2000.

[29] C. S. Wallace and D. M. Boulton. An information measure for classification. *The Computer Journal*, 11:185–195, 1968.

[30] C. S. Wallace and P. R. Freeman. Estimation and inference by compact coding. *Journal of the Royal Statistical Society*, 49:240–251, 1987. Discussion pages 251-265.

[31] G. K. Zipf. *Human Behaviour and the Principle of Least Effort*. Hafner, New York, 1949.