MDLChunker: a MDL-based Model of Word Segmentation

Vivien Robinet (vivien.robinet@imag.fr)
TIMC-IMAG laboratory, University of Grenoble, Domaine de la Merci
38706 La Tronche, FRANCE

Benoît Lemaire (benoit.lemaire@imag.fr)
TIMC-IMAG laboratory, University of Grenoble, Domaine de la Merci
38706 La Tronche, FRANCE

Abstract

This paper applies a MDL-based computational model of inductive learning to the problem of word segmentation. The main idea is that syllables are grouped into words as soon as this operation decreases the size of the overall representation of the data, that is the codelength of information. When exposed to a stream of artificial words, our model (MDLChunker) is able to reproduce Giroud & Rey (in press) effect: humans learn sub-words as well as real words at the beginning, but after a while they learn real words better than sub-words. In order to better mimic human learning, a limited-size short-term memory was added to the model and estimates of its size are given.

Keywords: inductive learning; word segmentation; minimum description length; computational model; distributional cues; simplicity principle

Introduction

In a seminal paper, Saffran et al. (1996) showed that, when exposed to a stream of concatenated artificial words, humans are able to segment correctly and learn the words from the only transitional probabilities between syllables. This result suggested that infants, although sensitive to acoustical factors such as phrasal prosody or lexical stress (Swingley, 2005), could be influenced by distributional cues to segment the stream of speech they are exposed to.

This paper presents a cognitive computational model (MDLChunker) to account for that phenomenon. It is based on the general idea that humans tend to make decisions leading to the simplest representation, i.e. minimizing the codelength of information in memory. Compared to existing models, it does not rely on any adjustable parameters.

Originally, MDLChunker was build to predict human performances on artificial grammar learning tasks, which is a classical paradigm in the implicit learning field (Servan-Schreiber & Anderson, 1990). Our purpose is to use the same model to account for word segmentation, which is the task traditionally studied in statistical learning. In this way, we follow Perruchet & Pacton (2006) who propose that implicit learning and statistical learning are two approaches of the same phenomenon.

Existing models

Two strategies can be used to model the way infants may solve the word segmentation problem (Swingley, 2005), which corresponds to two kinds of distributional-based computational models: those following a top-down approach, by inserting boundaries into continuous speech (bracketing strategy) and those using a bottom-up approach, creating new units by grouping frequent ones (clustering strategy)

Parser (Perruchet & Vinter, 1998) uses the clustering strategy. It maintains a list of weighted candidate words, which can be viewed as a mental lexicon. At each time step, Parser randomly selects between 1 and 3 units in the input to form a new candidate word. A unit is initially a syllable, but can become a longer group of syllables through an aggregative chunking process, provided that their weights are high enough. This bottom-up mechanism is intended to model the *perception shaping* phenomenon: what we already learned affects our perception. We do not view a new item as composed of elementary items if it has already been learned, we view it as a whole. For instance, “bkjbk” is viewed as a sequence of 5 letters whereas “obama” is viewed as one item. *Parser* contains a reinforcement parameter increasing the weight of the current percept. A forgetting mechanism is also implemented by a constant diminution of each weight at each time step. Finally, an interference mechanism slightly decreases the weight of each candidate word which shares a syllable with the new percept. When applied to a long sequence of concatenated artificial words, the best-weighted candidate words are those of the language (Perruchet & Vinter, 1998). BootLex (Batchelder, 2002) follows a bottom-up approach similar to PARSER.

Top-down word segmentation can be simulated by connectionist models. Christiansen, Allen & Seidenberg (1998) used a simple recurrent network to extract word boundaries from a corpus of child directed speech.

Brent & Cartwright (1996) model also follows a top-down approach. It does not contain a mechanism to aggregate the syllables, but only assess the relevance of a given segmentation. Therefore, it exhaustively generates all possible segmentations and uses the minimum description length principle (MDL) to select the most relevant one. Similarly, Argamon et al. (2004) uses the MDL principle in a segmentation task consisting in finding prefixes and suffixes inside words. Since our model also uses that minimum description length principle, we now present this idea.

A MDL-based model

Following Chater (1999), our hypothesis is that simplicity can account for many cognitive tasks. We already built a model (Robinet et al., 2008), that implement the general notion of simplicity using the formalism provided by the MDL principle. This model was designed to predict the time course of concept creation in a task where participants are learning an artificial grammar. We believe this model is general enough to apply to other tasks, such as word segmentation. When applied to the word segmentation task, MDLChunker could be seen as an online version of Brent & Cartwright (1996) model, with an explicit representation of chunks progressively updated over time.

From this point of view of simplicity, a good segmentation would minimize the amount of information which has to be stored. Using a lexicon would compress information by limiting redundancy. For instance, since the sequence of letters “o b a m a” is frequent, adding the word “w1: o b a m a” to the lexicon would compress the input. Adding “w2: b k j b k” would not compress anything, only consuming the memory size necessary to define it. A good segmentation can therefore be viewed as a trade-off between the conciseness of the lexicon and the expressiveness of the input data with respect to that lexicon. For instance, a very small lexicon, although saving resources, would lead to a high number of unrecognized words in the input. On the contrary, a very detailed lexicon corresponding to numerous combinations of syllables would take a large place in memory, although being good at processing new inputs.

Information theory offers a formal way to implement that idea, namely the minimum description length (MDL) principle (Rissanen, 1978). This method consists in computing the lengths of the codes for representing the lexicon (hereafter represented as Lexicon) and the lengths of the codes for representing the input data knowing the lexicon (hereafter represented as Data|Lexicon), and minimize their sum. Codelengths are estimated by means of Shannon's formula, saying that a symbol s, occurring with probability p, can be ideally compressed with a binary code whose length is $I(s)=-\log_2(p)$. In our case, p is estimated by the frequency of s.

Let us give an example. Suppose a language composed of the two words “abc” and “cb”. Input is therefore a long concatenation of these words. A good lexicon would contain “abc” and “cb” after enough data has been processed. Suppose that, so far, we have already processed “abcabcabcabccbcbabccbabcabcabc”. Because we have already learned the words “a”, “b”, “c”, “ab” and “cc”, the data was segmented as such: “ab c ab c ab c ab cc b c b ab cc b ab c ab c ab c”. If we need to process “abcab”, there are several ways to segment the new input. Let us consider two of them: “ab c ab” and “abc ab”. Case #1 only uses existing words whereas case #2 suggests “abc” as a new word. Figure 1 presents codelengths for the two cases.

Case #1:

Lexicon=a, b, c, ab, cc ; Data|Lexicon=ab c ab c ab c ab cc b c b ab cc b ab c ab c ab c ab c ab.
For example, codelength for c is $-\log_2(11/32)$ because c occurs 11 times out of 32. Codelength(cc) is $-\log_2(3/32)$, etc. The size for the entire case #1 is therefore 65.5 bits.

Case #2:

Lexicon=a, b, c, ab, cc, abc. Since a new word “abc” is added to the lexicon, the data are segmented accordingly: Data|Lexicon=abc abc abc abc c b c b abc c b abc abc abc abc ab
The total size for case #2 is 63.4 bits. Therefore, case #2 is a better segmentation because its total size is smaller.

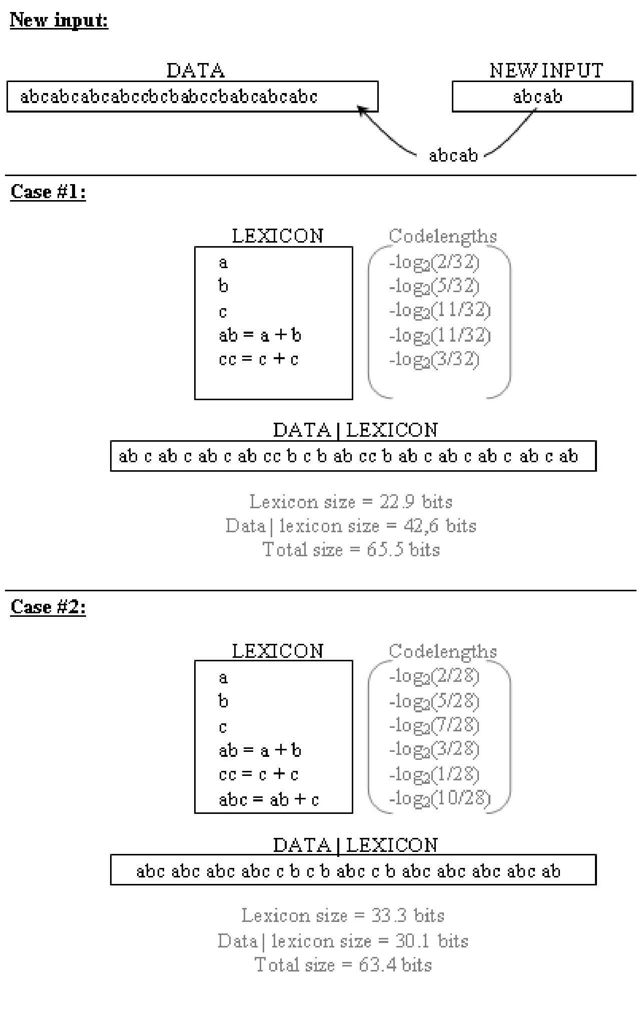


Figure 1: Sizes of lexicon and data for case #1 and case #2 after processing the input “abcab”

This way of selecting the most probable segmentation humans have chosen can be viewed as a form of Occam's razor: given several segmentation hypotheses, humans would select the simplest one. This idea that simplicity

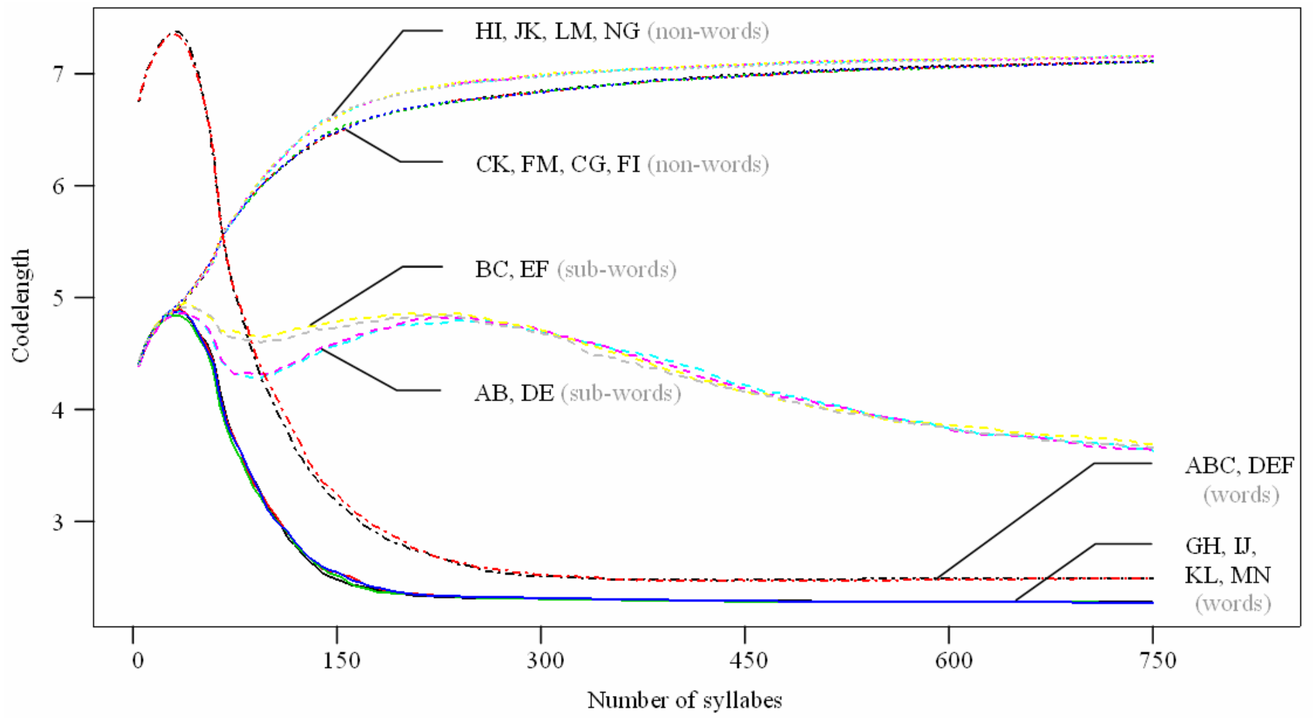


Figure 2 : Time course of codelengths obtained with our model during the processing of the first 750 syllables. Codelengths are presented for all the words, sub-words and non-words of the experiment.

could be a unifying principle in cognitive science has been discussed by Chater & Vitanyi (2003).

As we mentioned earlier, Brent & Cartwright (1996) used the MDL principle for selecting the best segmentation, but they had to generate all possible combinations. In order to be more cognitively plausible, we designed a model to account for the process by which words are progressively aggregated through the process of the input.

In its first version, MDLChunker considers a fixed-size part of the input at each step. It then operates in 3 sub-steps:

- the new piece of input is encoded using the existing words in the lexicon;
- the model creates new words by grouping two existing words, provided that they decreases the overall size (MDL principle);
- data is re-expressed using the existing words.

With this architecture, our model is able to correctly segment a stream of artificial words, even if words share some syllables. It successfully learned the artificial words of the Perruchet & Vinter (1998) experiment.

The vanishing sub-word effect

This model was also tested on data from Giroux & Rey (in press). They designed a new experiment to compare recognition performances of adults hearing either 2 or 10 min of an artificial spoken language. The language is composed of 6 words, formally represented here as ABC, DEF, GH, IJ, KL and MN. Participants just listened to a random concatenation of these words, uttered at the rate of 3.3 syllables/s by a speech synthesizer without any prosodic information.

After a 10 min training, recognition performances on dissyllabic words (such as "GH") was significantly higher than those obtained on dissyllabic sub-words (such as "BC"), while no difference occurs after a 2 min training.

Authors successfully reproduced this vanishing sub-word effect with PARSER, using the same Perruchet & Vinter (1998) parameter values. This result credits the bottom-up hypothesis by suggesting that sub-word recognition is a necessary step in the word learning process.

Our model was run on the same artificial language to check whether we could reproduce this vanishing sub-word effect. The input was split in blocks of 5 syllables[1]. After each block was processed, codelengths were computed for the 2 tri-syllabic words (ABC and DEF), the 4 dissyllabic words (GH, IJ, KL and MN), the 4 sub-words (AB, BC, DE and EF) and the 8 non-words used by Giroud & Rey (CK, FM, CG, FI, HI, JK, LM and NG). 1000 simulations were performed with random input sequences and codelengths were averaged.

Figure 2 presents all codelengths as a function of the number of syllables processed. The lower the codelength,

[1] Because input was split in blocs of 5 syllables, trissyllabic words are statistically broken more often than dissyllabic ones. Thus, isolated syllables from trissyllabic words are more frequent and have a lower codelength

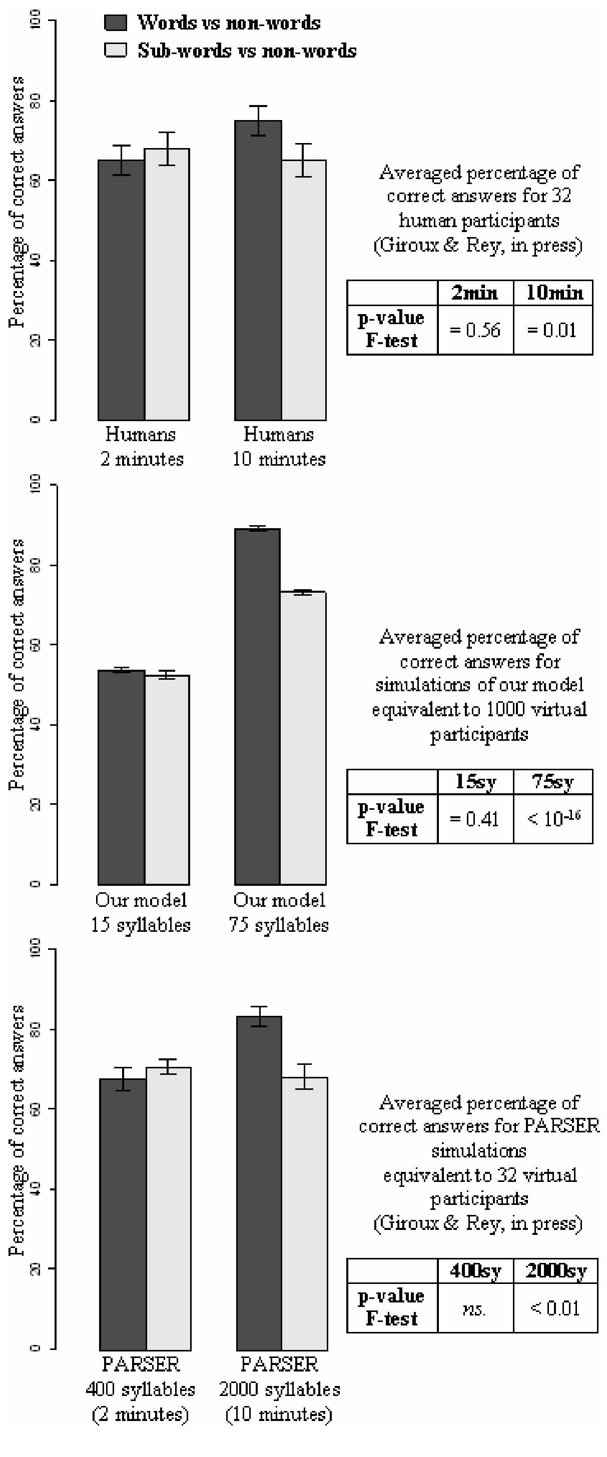
the more frequent the word. At the beginning, there is no difference between all dissyllabic words and sub-words, but tri-syllabic ones are trivially coded using 50% more space. After about 30 syllables have been processed, things begin to change: non-words[2] costs more and more to be represented, whereas real words, both dissyllabic[3] and trissyllabic, are efficiently coded, even less than sub-words.
We used the test designed by Giroux & Rey, in order to compare our model to human data. The words GH, IJ, KL, MN are tested against the non-words CK, FM, CG, FI and the sub-words AB, DE, BC, EF are tested against the non-words HI, JK, LM, NG. In this test, the model has to choose the dissyllabic unit that best matches with the language on which it was trained (in our case, that with the lowest codelength). The two tests (words vs non-words and sub-words vs non-words) are performed eight times for each training phase (virtual participant). Averaged performances are presented Figure 3 along with those obtained by Giroux & Rey.

No significant difference between performance on words over sub-words was obtained with our model at 15 syllables (F(1, 999)=0.69 ; p=0.41), whereas a significant difference (F(1, 999)=465 ; $p<10^{-16}$) is observed at 75 syllables.
Fed with the same artificial grammar, our model could thus reproduce this vanishing sub-word effect, but the rate of learning appears too fast compared to the human counterpart. This is due to the fact that our model never forgets anything. In order to be more cognitively plausible, we improved our model by restricting its memory i.e. the data from which the model could make associations to create new words. We also changed the way the input is processed in order not to split the input into fixed-size parts but instead to process it as a stream of syllables.

Improvement: adding of a memory module

Instead of taking into account all the data already processed, this new version of the model used a memory buffer to only keep a given amount of data. This buffer plays the role of a short-term memory (STM) whereas the set of existing words (the lexicon) is more like a long-term memory. The model works in the following way (Figure 4):

- The beginning of the current input is segmented in order to minimize its codelength and the first two units are candidates for forming a new word (perception shaping);



	2min	10min
p-value F-test	= 0.56	= 0.01

	15sy	75sy
p-value F-test	= 0.41	$< 10^{-16}$

	400sy	2000sy
p-value F-test	*ns.*	< 0.01

Figure 3 : Percentages of correct results for humans, PARSER and our model, on the two tests: words vs non-word and sub-words vs non-words. A 2 minutes training corresponds to 400 syllables and a 10 minutes training to 2000 syllables.

[2] A difference between HI, HK, LM, NG and CK, FM, CG, FI could be observed in Figure 2, because of the lower codelength of syllables C and F which are parts of trissyllabic words (see footnote 1).

[3] A difference between BC, EF and AB, DE could be observed in Figure 2. While our model systematically creates the first among two possible new words, AB and DE are created more often than BC and EF. The former are only created if the input split breaks ABC in A and BC, which is less frequent than having either ABC or AB and C.

- This new word is created only if its creation would decrease the overall size of the system (STM rewritten + lexicon with the new word).
- The new percept (the first input unit) is added to STM;
- Old percepts exceeding memory size are removed from STM. This step depends both on the memory size and on the codelength of the oldest percepts.

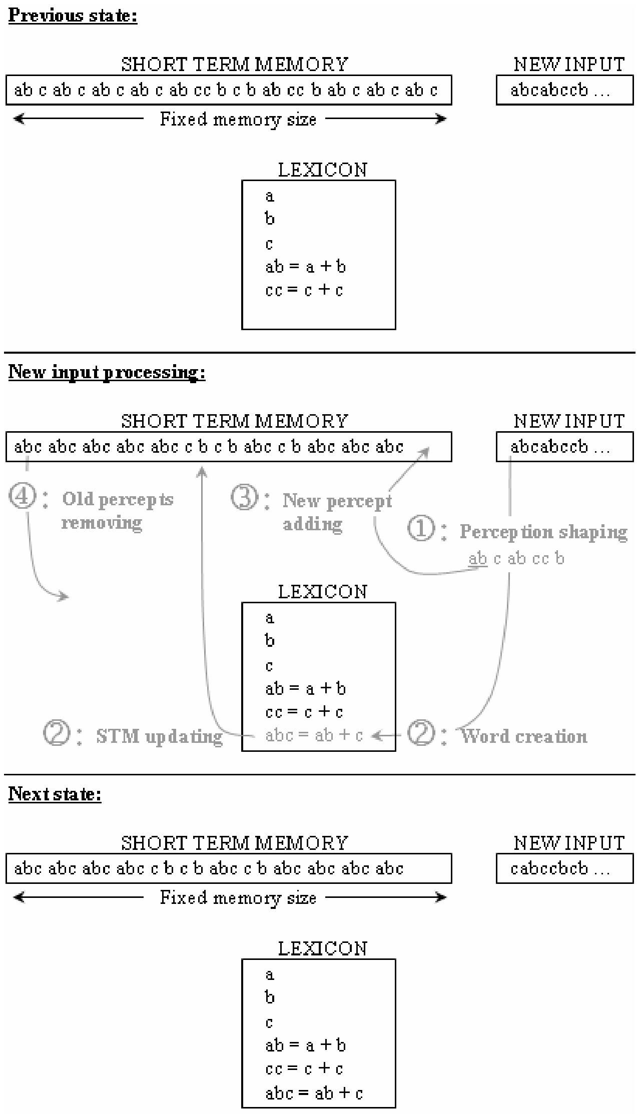


Figure 4: Architecture of our system after adding of the memory module

Let us go back to the previous example in order to highlight the changes in the new version of the model. Existing words are “a”, “b”, “c”, “ab” and “cc”. Previous STM state is the following: “ab c ab c ab c ab c ab cc b c b ab cc b ab c ab c ab c”. New input is “abcabccb ...” shaped as “ab c ab cc b” (Figure 4, step 1). The first two units are selected in the new input. In our case, this is “ab” and “c”. The new word “abc” is then candidate for being a new word in the lexicon (Figure 4, step 2).

With this new word, the state of the system would be represented using 63.4 bits, which is better than the 65.5 bits if “abc” were not created. Therefore, the new word is created and “ab” is added to STM (Figure 4, step 3).

There are several ways to avoid STM overflow when a new set of syllables has to be added. We could have kept a fixed number of “words”. In order to be more coherent with the rest of our model, we kept a fixed quantity of information i.e., a maximum number of bits. After a new percept has been added to memory, old percepts are therefore removed to keep memory size under its limit. Then the process continues with the next input.

Estimation of a good memory size

By supplementing our model with a finite memory, we added a parameter (the memory size) that needs to be adjusted. With a huge memory (1000 bits), the model learns at a very high rate, like in its previous version. This is not cognitively plausible and does not correspond to human data. With a too small memory (100 bits), no learning occurs at all, because there is not enough data available at the same time in memory to find regularities. With a size of 150 bits, we found a good similarity between the model and the human learning rate. With this 150 bits memory size, we reproduced both the vanishing-words effect, and the time course of learning. At 2 minutes (400 syllables) no significant difference ($F(1, 999)=-0.21$; $p=0.64$) was observed on performances between words and sub-words. This difference became significant ($F(1, 999)=48.4$; $p<10^{-12}$) after a 10 minute training.

Comparison with PARSER

Our MDL based model obtains results that are very close to those obtained with PARSER. These two chunking models use two different approaches to implement the same functions: perception shaping, reinforcement, forgetting and interference.

Perception shaping

The underlying idea is that perception depends on what was already learned.

PARSER: Shape the input by selecting the longest existing unit above a fixed threshold (shaping threshold).

MDLChunker: When looking for the shortest encoding of the input, the model tends to prefer units with shorter codelengths, since codelength depends on earlier perceptions.

Reinforcement

The idea is to increase the weight of recently perceived unit in order to give frequent units a higher weight.

PARSER: Add a fixed weight to the perceived unit. The weight differs depending on the position relative to the shaping threshold.

MDLChunker: Perception of a new unit increases its frequency, thus naturally decreasing its codelength.

Forgetting

The idea is to decrease the weight of the units that no longer occurs.

PARSER: At each iteration it decreases all the unit weights by a fixed quantity.

MDLChunker: Frequency of unperceived units slightly decreases naturally after each new perceived unit.

Interference

The idea is that when shaping the perceived unit by using some units, other units that could apply have their weight decreased.

PARSER: Decrease the weight of all the units overlapping the perceived unit.

MDLChunker: Frequency of units not used to encode the perceived unit slightly decreases naturally, which in turn increases its codelength, thus making it less interesting for future perceptions.

Conclusion

In this paper we show that, our original chunking model designed for an implicit learning paradigm, can easily account for word segmentation, which is the classical paradigm used in statistical learning (Saffran et al., 1996).

Without any improvement in our parameter-free model, we successfully reproduced the Perruchet & Vinter (1998) experiment as well as the vanishing sub-word effect presented by Giroux & Rey (in press). We observed no significant differences between words and sub-words at the beginning of the training, following by a significant superiority of words over sub-words during the training. While the memory size is infinite in this first version of our model, the learning rate is much higher than humans one. We limited this memory size to account for forgetting effects. We found that for a size of 150 bits, model and humans learning rates are equivalent according to the conducted tests.

Both MDLChunker and PARSER can extract words from a stream of syllables. They are also able to reproduce the vanishing sub-word effect, suggesting that chunking is an efficient bottom-up process to model word segmentation.

We plan several kinds of improvements in the future. First, this general MDL-based model of inductive learning needs to be applied to other word segmentation tasks as well as other problems. We also aim at improving its model of memory and we will probably attempt to unify the representation of the lexicon (long-term memory) and the data already processed (short-term memory). The idea that memory could be modeled in term of quantity of information (Brady et al., 2008) appear challenging to us. It would be interesting to see how this point of view compares with the classical view of expressing memory limit in terms of a number of chunks (Miller, 1956).

References

Argamon, S., Akiva, N., Amir, A., & Kapah, O. (2004). Efficient unsupervised recursive word segmentation using minimum description length. In *Proc. 20th International Conference on Computational Linguistics (Coling-04).*

Batchelder, E. O. (2002). Bootstrapping the lexicon: A computational model of infant speech segmentation. *Cognition, 83*(2), 167-206.

Brady, T. F., Konkle, T., & Alvarez, G. A. (2008). Efficient Coding in Visual Short-Term Memory: Evidence for an Information-Limited Capacity. In B. C. Love, K. McRae, & V. M. Sloutsky (Eds.), *Proceedings of the 30th Annual Conference of the Cognitive Science Society* (pp. 887-892). Austin, TX: Cognitive Science Society.

Brent, M. R., & Cartwright, T. A. (1996). Distributional regularity and phonotactic constraints are useful for segmentation. *Cognition, 61*(1-2), 93-125.

Chater, N. (1999). The Search for Simplicity: A Fundamental Cognitive Principle? *The Quarterly Journal of Experimental Psychology A, 52*, 273-302.

Chater, N., & Vitanyi, P. (2003). Simplicity: a unifying principle in cognitive science? *Trends in Cognitive Sciences, 7*(1), 19-22.

Christiansen, M. H., Allen, J., & Seidenberg, M. S. (1998). Learning to segment speech using multiple cues: A connectionist model. *Language and Cognitive Processes, 13*(2/3), 221-268.

Giroux, I., & Rey, A. (in press). Lexical and sub-lexical units in speech perception. *Cognitive Science*.

Miller, G. A. (1956). The magical number seven, plus or minus two: Some limits on our capacity to process information. *Psychological Review, 63*(2), 81-97.

Perruchet, P., & Pacton, S. (2006). Implicit learning and statistical learning: one phenomenon, two approaches. *Trends in Cognitive Sciences, 10*(5), 233-238.

Perruchet, P., & Vinter, A. (1998). PARSER: A Model for Word Segmentation. *Journal of Memory and Language, 39*(2), 246-263.

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

Robinet, V., Bisson, G., Gordon, M., & Lemaire, B. (2008). Modèle cognitif de l'apprentissage inductif de concepts. In *Actes du colloque annuel de l'association pour la recherche cognitive*. France.

Saffran, J. R., Aslin, R. N., & Newport, E. L. (1996). Statistical Learning by 8-Month-Old Infants. *Science, 274*(5294), 1926.

Saffran, J. R., Newport, E. L., & Aslin, R. N. (1996). Word Segmentation: The Role of Distributional Cues. *Journal of Memory and Language, 35*(4), 606-621.

Servan-Schreiber, E., & Anderson, J. R. (1990). Learning Artificial Grammars With Competitive Chunking. *Journal of Experimental Psychology: Learning, Memory, and Cognition, 16*(4), 592-608.

Swingley, D. (2005). Statistical clustering and the contents of the infant vocabulary. *Cognitive Psychology, 50*(1), 86-132.