PARSER: *User Manual*

Perruchet, P., & Vinter, A. (1998). PARSER : A model for word segmentation. *Journal of Memory and Language*, 39, 246-263.

This software is composed of 3 files:

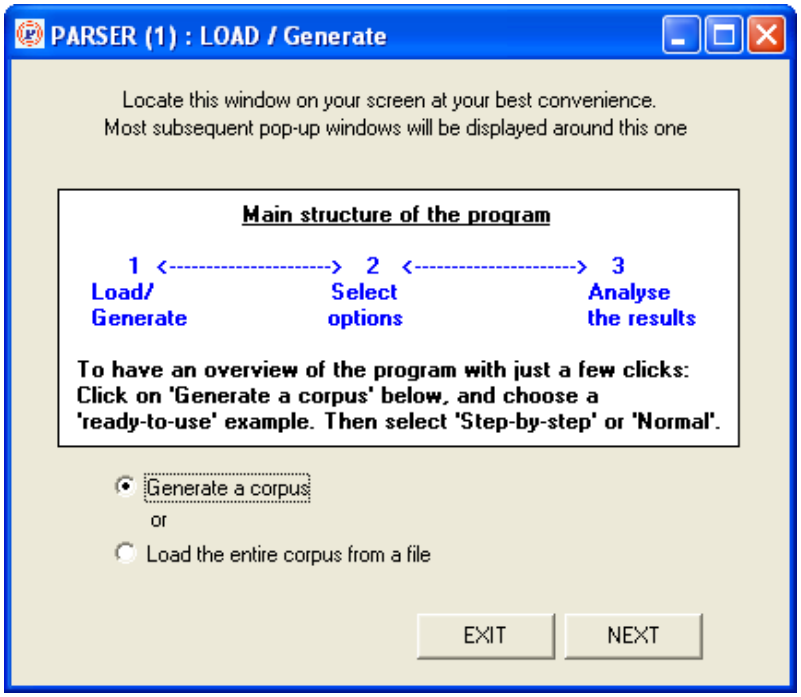
- P98pilot6.exe, which is the program the user has to start and
- SP98pan2.exe
- Mkcorpus.exe, which are subprograms that are called by the main program.

These files must be *imperatively located in the same folder*. There is no other requirement for the installation. However, creating a specific directory for these files may be a good practice. Indeed, the program creates a few temporary files in this directory. They are normally deleted, but if the program fails to reach its end for any reasons, some of these files may remain on the hard disk (note that removing residual temporary files is not necessary: they will be deleted in subsequent simulations).

In principle, Parser is relevant (and hence the program is applicable) to any issue in which a sequential material needs to be segmented into units that are not given in the sensory input. The primitives may be, for instance, phonemes, letters, or geometrical figures. However, Parser has been mainly applied to the issue of discovering the words from a continuous speech flow, with the syllables construed as processing primitives. For the sake of simplicity, the terminology relevant for the word segmentation literature will be used here.

For a first appraisal

To have a quick overview of the program, run 'P98pilot6.exe', and select '*generate a corpus*' on the window below. The next window is designed to enter the materials the program needs to generate a corpus. Select one of the *Ready-to-use examples* and the program will fill the form for you. The left-hand panels comprise the items that will be concatenated to build the corpus with their respective frequency. The other two panels comprise the items used for the test.



After an example has been selected, the next pop-up window presents the main options. The only mandatory choice is between '*Step-by-step*', which provides a detailed analysis on a single run, and '*normal*'. Click on '*Step-by-step*', then "*START*". You first see a summary of the current set-up and the corpus that the program has generated. The next window is the results window.

The results window comprises two main frames, which shows the results for Steps 1 and 2, respectively. This mode of presentation is intended to make easier the analysis of the operations performed by the model on each step. The part of the currently processed corpus is displayed on the top of the page.

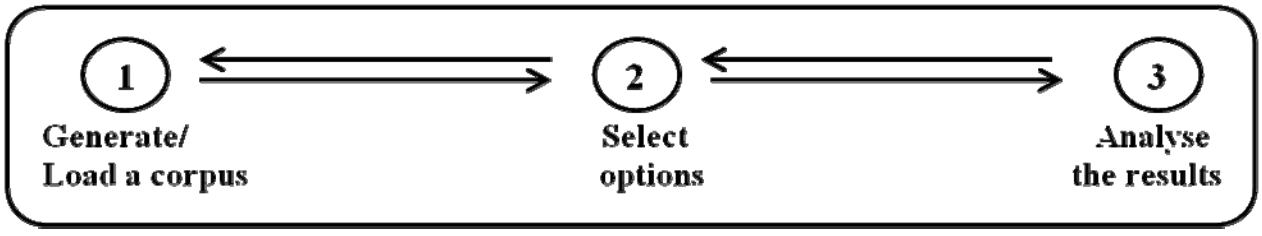
A detailed comment on how the model works is available on this web site:
http://leadserv.u-bourgogne.fr/LEAD/people/perruchet/soc.html

The two bars in the high-right corner indicate the standard scores of completeness (the proportion of words that are extracted) and precision (the proportion of actual words among the extracted units), respectively. The program also returns the number of discovered items belonging to each list (here: test words and test part-words), and the number of items that have been found but which do not belong to the list(s). The remaining information (mainly: the content of the Percept Shaper) should be self-obvious for anyone familiar with the model.

The program automatically stops on the first window, in order to let it up to you how to go through the next steps. Standard media buttons (except there is no 'rewind' function) allows to go ahead either step-by-step, or in a running way. Shifting between the two modes is obviously possible. At any moment, the right-hand frame displays the result of the current step, and the left-hand frame displays a record of Step N-1 for the sake of comparison.

You may be concerned with the slowness of this analysis. Don't worry: there is another mode of functioning, the "*Normal*" mode. Click on "*Back to 'select option'*", and select now '*Normal*' instead of *Step-by-step*, then validate this choice and the *Summary/ Corpus* sheet. The final result of the analysis now appears directly on the right-hand panel of the results window. If you have previously let the *step-by-step* analysis running until its end, you may see that the two sets of results are different. This is because the random seed is set to 'Time' as a default. Had the random seed been set to a fixed value, say 1, the two modes (*Step-by-step* and *Normal*) would have returned exactly the same values (see more on the random seed below).

Besides the advantage of speediness, the *Normal* mode makes various options available. The most useful may be the *number of runs*. Enter a value, say 5, in the "*Number of runs*" combo box, and let the results scrolled up on the screen. The left-hand panel of the results window now displays a summary of the results for the 5 runs. The program is described in more detail below.



The main structure

1 - The opening window: Generate vs. Load a corpus

You have two main options to enter the data: either you provide the basic material to the program (i.e., at a minimum, a list of words and the number of occurrences of each word) via the keyboard (or by copy/paste from Word or Excel) and the program will generate a corpus, or you load a complete corpus that you have previously created (with another software) and saved as a text file.

Data coding in Parser: General principles

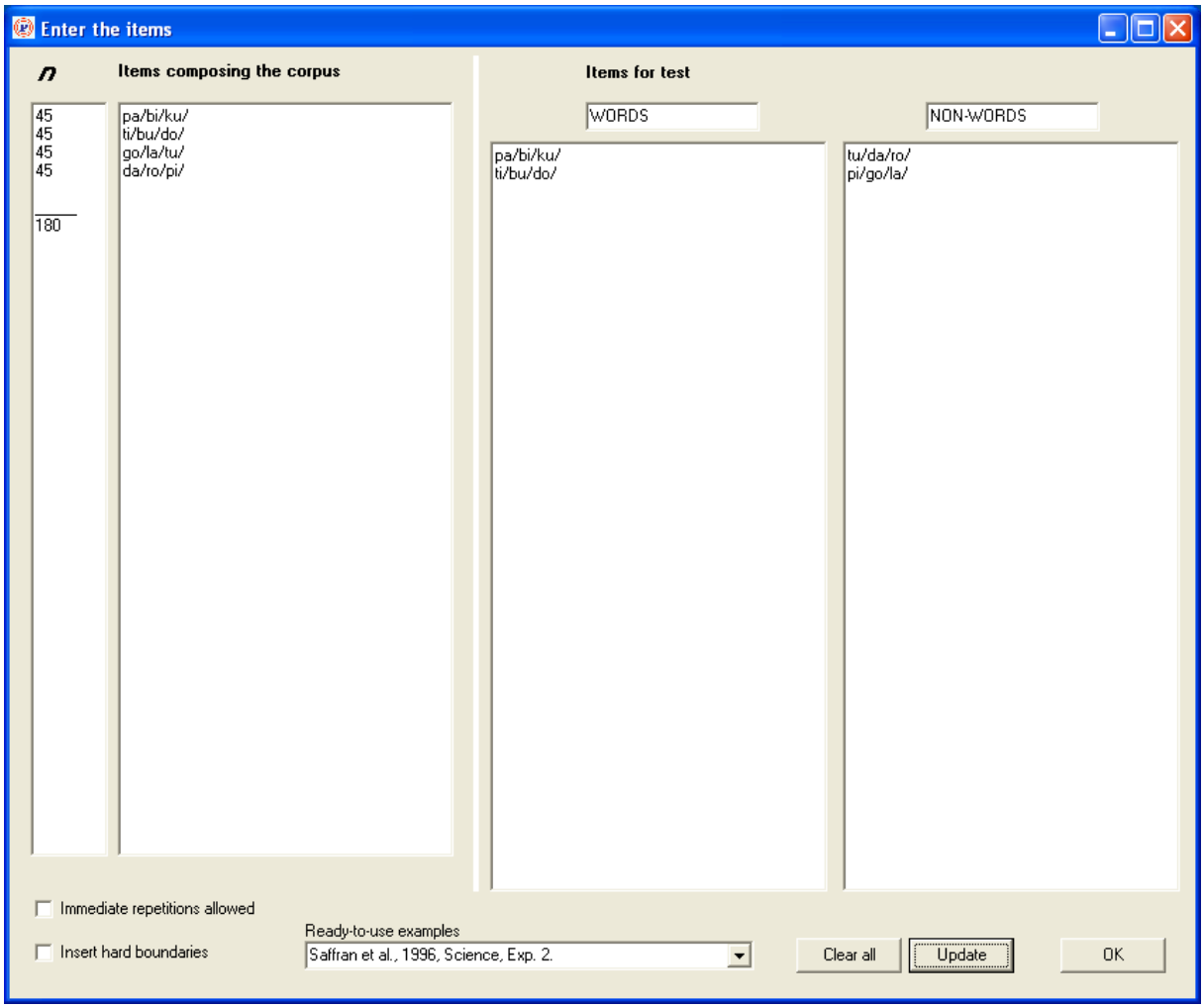
To process a string of characters, the program needs to know the boundaries of the primitives, i.e., of the strings of characters that are considered as indivisible units. For instance, if syllables are considered as primitives for a given analysis, the letters composing a syllable will be ever processed as a whole. In Parser, a primitive ends with a '/'. A primitive may comprise a variable number of characters.

Generate a corpus

You need to enter the ***items composing the corpus*** in the left panel of the window below. Entering test items (in the two right-hand panels) is optional. Whatever the list, the coding is the same. The items must be segmented in (indivisible) primitives, which end by '/' (but see the note below) An item may comprise a variable number of primitives, and a primitive may comprise a variable number of characters. E.g.: '1/6/3/', 'b/u/p/a/d/a', 'bu/pa/da/..' , 'par/ti/ci/pant/' are legal items.

You also need to enter the *number of repetitions* of each item. If this number is common to all items, it suffices to enter the target value at the top of the most left-hand column (clicking on '*update*' is optional, but ensure you that the frequencies have been set as you want). More generally, any line left empty in the frequency column will be recursively completed with the value immediately above. As you may observe if you scan through the *ready-to-use examples*, the number of repetitions is constant in most studies, but it differs in others. When the frequency of the items differ, and no immediate repetition is allowed, the usual algorithms of randomization provide a flawed outcome (see French & Perruchet, Behavior Research & Method, 2009). The program uses an algorithm derived from the one proposed by French & Perruchet, which ensures that each item occurs exactly the number of times that has been required, with an homogeneous distribution throughout the corpus.

As a default, the words will be concatenated randomly without immediate repetition to form a continuous corpus. However, it is also possible to authorize *immediate repetitions* by checking the appropriate box. As you may observe if you scan through the *ready-to-use examples*, some studies prohibited immediate repetitions while other did not. It is also possible to process the corpus as a succession of separate sentences, hence making language exposure a bit more natural. To do that, check the box '*Insert hard boundaries'*, and complete the pop-up form with either a fixed value (the number of words composing a sentence), or a range of values.



In addition, two lists of ***test items*** may be entered. The test items must be in the same format as the language, i.e. written with a '/' after each primitive. The two lists are, by default, labeled 'words' and 'part-words'. However, any other assignment is possible (and the labels can be edited accordingly). For instance, if you select *Giroux & Rey (2009)* in the *ready-to-use examples*, you see that the labels have been changed, because the crucial comparison here is between words and sublexical units (instead of part- words). Note that entering non-words (i.e., a sequence of syllables not displayed in the corpus) is objectless: Parser cannot create non-words. When test list(s) are provided, the program returns various scores (see below)

The number of items is usually limited, so entering the data via the keyboard should be a manageable task. However, it may be more convenient to copy/paste the items from a file.

Load a corpus

Although the '*Generate*' option allows to simulate a large part of the word segmentation literature, there are also obvious limits. For instance, one may hope to introduce novel words progressively across training. This kind of manipulations is not possible under the '*Generate*' mode.

Before selecting the '*Load a corpus*' option, you have to prepare a text file containing the entire corpus with another software. The text must be segmented into primitives, which are separated by '/', as specified above. The program also needs to know if the corpus can be considered as a continuous sequence of primitives, or if there are hard boundaries. Hard boundaries separate physically

discontinuous utterances – No unit straddling over a hard boundary will be created. In Parser, the hard boundaries are coded with '//'.

Paragraph marks, spaces, and the following punctuation symbols: . , ! and ? can be included for user's convenience, but they have no function at all: only '/' and '//' are recognized as separators. Any other character (more precisely: any character the ASCII code is comprised between 34 and 255, with the exception of the punctuation marks listed above, and, of course, '/') is coded as an element of a primitive.

For example:

(1) this/is/the/first/sen/tence//this/is/the/se/cond/sen/tence/

(2) this / is / the / first / sen/tence,/
/this / is / the / se/cond / sen/tence/.

and

(3) this/
is/
the/
first/
sen/
tence//
this/
....

are equivalent: in all cases, syllables are primitives and the two sentences are separate utterances.

By contrast, note that

(1) this/

and

(2) This/

are different primitives, because 't' and 'T' are coded as different characters. Although this choice appears to be rather inappropriate in this specific case, lower-case and upper-case letters are considered as different characters due to their distinctive function in the phonetic code.

After having loaded a file, the user is offered the possibility of entering the words of the language and/or test words. This information is obviously ignored by the program during the extraction process, but, if provided, it is exploited for analyzing the results. The procedure is the same as the one described above ('*Generate a corpus*'), except that a few irrelevant options are made inactive.

Note: In many cases, the primitives can be coded with a single letter or digit, and this coding has the advantage of making processing faster (long primitives slow down the processing). For convenience, a general convention is that if there is no single slash in a string (e.g., in the whole corpus, or in a test item), the individual characters composing the string are taken as primitives.
In a nutshell: NO SLASH = SLASHES ANYWHERE
For instance:

(1) abc

(2) a
b
c

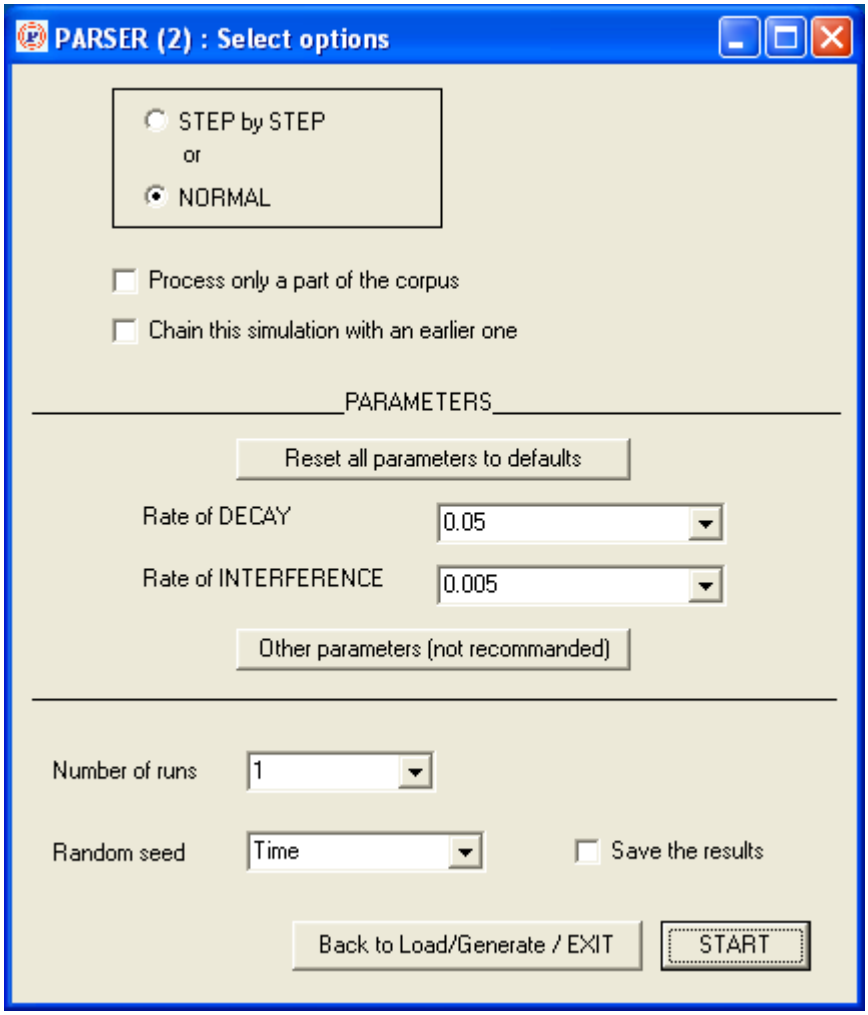
and

(3) a/b/c/

are valid and equivalent entries.
In the corpus, '//' can still be used to indicate hard boundaries.

2 - Select options

After the data have been entered, the next window to appear is represented below. It is recommended to select first whether you are interested in exploring in detail a single simulation (*Step-by-step* mode), or if you intend to perform one or several simulations in a relatively efficient way (*Normal* mode). Indeed, this choice determines whether other options are in active or inactive state (many options are inactive pending you click on '*Step-by-step*' or '*Normal*'). If you select '*Normal*', be sure that the subprogram 'SP98pan2.exe' is in the same folder as the main program. 'SP98pan2.exe' is MS-DOS program, which is called by the main program under the '*Normal*' mode.



Process only a part of the corpus

(Available only when the corpus has been loaded from a file –The number of repetitions is a parameter in the '*Generate*' mode). If you have prepared a file with a long corpus, and that you wonder about the model's performance with a smaller corpus, you don't need to prepare and save a new file. When the '*process only a part of the corpus*' box has been checked, you are asked how many primitives you wish to keep for the next analysis. Learning curves can be drawn in this way..

Chain the simulation with an earlier one

As pointed out below, results can be saved. To start a new simulation in the state reached after a previous one, it suffices to indicate to the program the file in which the previous results have been saved, through a standard Windows dialog box. This should be especially useful when the model has to be exposed to several languages in succession. Let us suppose that you want to introduce new words progressively during training. You can prepare a complete corpus and load it as a whole, but it is also possible to create different corpuses with the '*Generate'* option, and to chain the simulations.

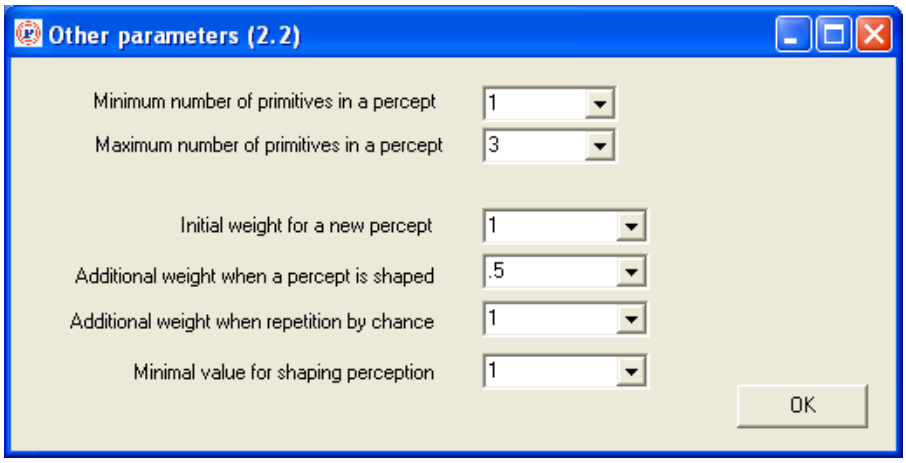
To use this option, the results file must have been left unchanged. More precisely, it is possible to add information to the initial part of the result file, in which the set-up of the current simulation is reported, as well as to the final part, which reports a summary of the scores, but strictly no change (including the addition or removal of simple spaces) is allowed for the results corresponding to each run (i.e., to the sections reported between two lines composed of "==========...")

Setting the parameters

Parser has been occasionally criticized on the ground that the model includes a large number of parameters (e.g., Bonatti et al., 2006). This criticism is unwarranted, for at least two reasons. Firstly, most parameters have a clear psychological meaning, which is certainly not the case of all parameters in modelisation studies (e.g., the meaning of the momentum in connectionist networks is somewhat opaque). Secondly, Parser has proven to be remarkably insensitive to parameter manipulations. The skeptics are invited to run simulations with the *Ready-to-use examples* provided in the input window. Anyone can see that Parser performs rather well in all cases. Now, these examples are drawn from various studies coming from various laboratories, they manipulate several variables (frequency, transitional probabilities, length of words, etc.), and there is no specific adjustment: the model all simply processes on-line the material displayed to the human participants (e.g., the length of the processed corpus is the same). Likewise, no parameter fitting is required. Worthy to note, the parameters that have been set as a default in the program have not been selected for this specific use: they are those used in the first paper on the model (Perruchet & Vinter, 1998), and which have been used in most subsequent papers.

That said, it may be useful to adjust some parameters in some occasions. The program allows to modify all of them. The most important are the *rate of decay* and the *rate of interference*. Two main guidelines have to be kept in mind when these parameters are modified. First, the rate of forgetting (decay/interference) needs to be set at an intermediary value. If forgetting is too strong, the program fails to build any units, hence generating a low score of completeness. If forgetting is too low, the program stores a very large number of units, hence generating a low score of precision. Usually, running the *step-by-step mode* allows to find appropriate values without running complete simulations. Second, manipulating forgetting through the decay parameter makes the model essentially sensitive to frequency, while manipulating forgetting through the interference parameter makes the model essentially sensitive to transitional probability and contingency (for an explanation, see for instance Perruchet & Pacton, 2006).

The other parameters are displayed in the window below. The number of primitives composing a single percept may be conceived of as something like a working memory span, and it may make sense to adjust the values in some studies (e.g. in developmental investigations). The other values defines the way the weight of the units are incremented. Admittedly, these values are set arbitrarily, but the problem is more apparent than real. Indeed, what is relevant is the ratio between the increments (due to the on-line processing of the units) and the decrements (due to forgetting). For the sake of between-studies comparisons, it is advisable to left these values unchanged, and to manipulate the relevant ratio by changing what has been coined here as the main parameters, namely the rates of decay and interference.



If you have changed one or several parameters, it is possible to *reset all parameters to their default values* by clicking on the appropriate button. Note that this button is active only if one or several parameters have been changed during a prior simulation (as a consequence, seeing the button inactive means that the configuration of parameters is standard).

Number of runs

(Available only in the '*Normal*' mode. Run is set to 1 in the *Step-by-step* mode).
When the number of runs is >1, the user is asked whether the same corpus must be used for all runs, or if a new corpus must be used for each run. Indeed, the result from a single simulation may depend to some extent on some particularities of the specific corpus on which the simulation is performed. For instance, it is possible that a given word appears more frequently as expected by chance at the beginning of the corpus. This potential bias, although certainly a very minor source of biases (especially with long corpuses), can be prevented by asking a new corpus for each simulation.

When the corpus is generated by the program, this option involves no other constraint. Note, however, that generating a new corpus for each run unavoidably slows down the program. The time required to generate a corpus depends on the length of the corpus, but also on other conditions. As a rule, generating a corpus with words of different frequencies without immediate repetition may be very time-consuming.

When the corpus is loaded from a file, this options implies that the user has prepared a number of different files at least equal to the number of runs. The name of the first file (the file selected in the '*Load*' option) must be suffixed with '1' (e.g., data1.txt), and the subsequent files must be suffixed with '2', ..., 'n'.

Random seed

There are four main options:

- Selecting '*Time*' uses the computer clock as a seed. This ensures a different randomization in each case. However, a problem with this option is that it does not allow to reproduce the same set of events. Reproducing the same set of events may be desirable for various purposes. For instance, if one wishes to draw learning curves by entering increasingly long corpuses, reproducing the same events across successive simulations appears appropriate. The next options make it possible.

- The option '*CurrentRun*' uses the number of the current run as the random seed (i.e., 1, 2, ... n, in succession). If you have selected this option during training, and you are puzzled by the results

reported for, say, Run 7, you can enter '7' as the random seed under the *'Step-by-step*' mode to examine what happens in this particular situation.

- The option *'CurrentRun + c*' is identical, except that a constant is added to the number of the current run. This allows to obtain several sets of reproducible simulations (but be aware that entering '3' as a constant, for instance, will generate the same values for the first run as for Run 4 under the '*CurrentRun*' option.)

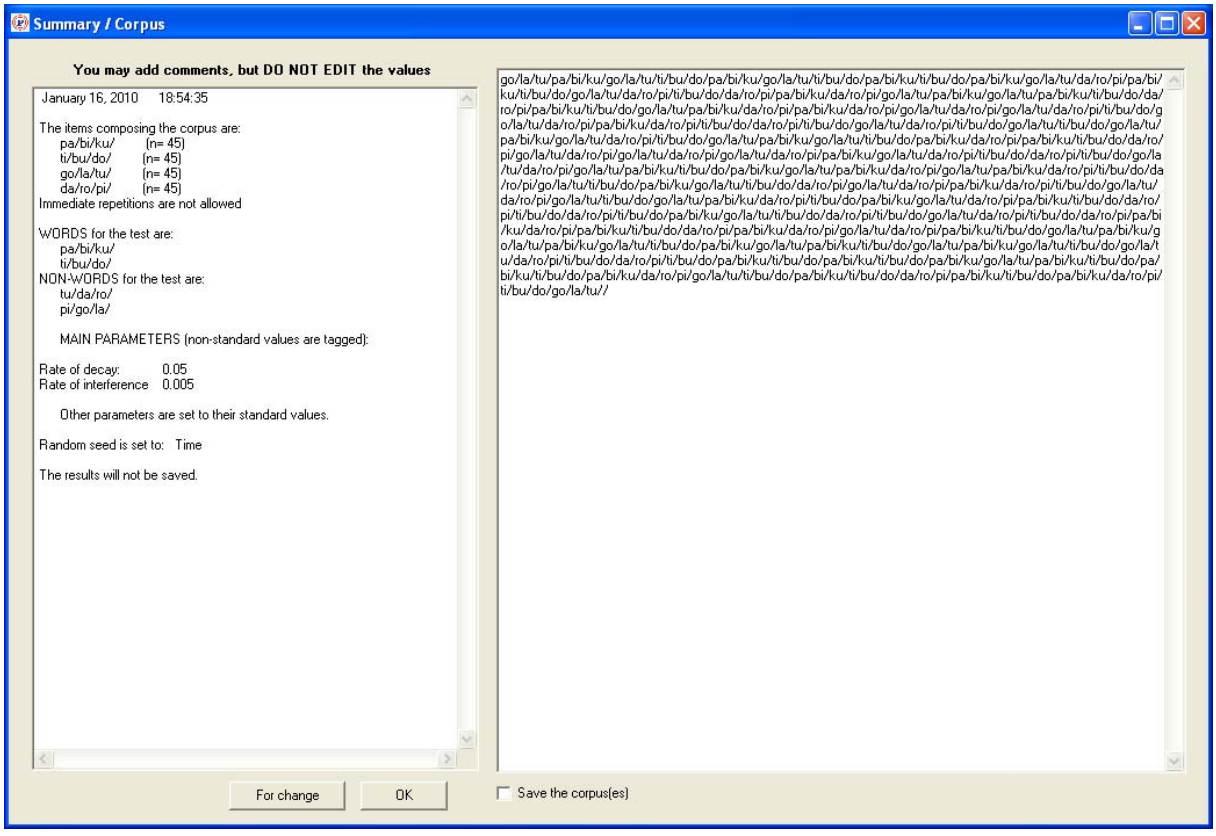
- Entering a numeric value (which needs to be an integer) allows to generate the same set of values each time this value will be selected again, but if you intend to reproduce Run 7 after having selected this option, you need to start again from Run 1. In addition, it is impossible to reproduce the sequence in the *'Step-by-step'* mode, because it is not possible to know the values serving as random seed for Run 7.

Save the results

(Available only in the '*Normal*' mode.). When this option has been selected, a complete record of the session is saved. The saved file includes: (1) the summary file, presented below, which recapitulates the whole set-up, (2) the complete results for each run and (3) a final table displaying the scores.

Summary

The '*Summary/ Corpus*' window pops up automatically before the analysis begins. It allows to check that everything has been set as you intended to do. It is possible to add further comments for your records on the *summary* form (note that the date of creation is automatically inserted). However, changing the target values (e.g., the rate of decay) on this form has no other effect than getting you in a mess! To change anything, click on *'For change'*.



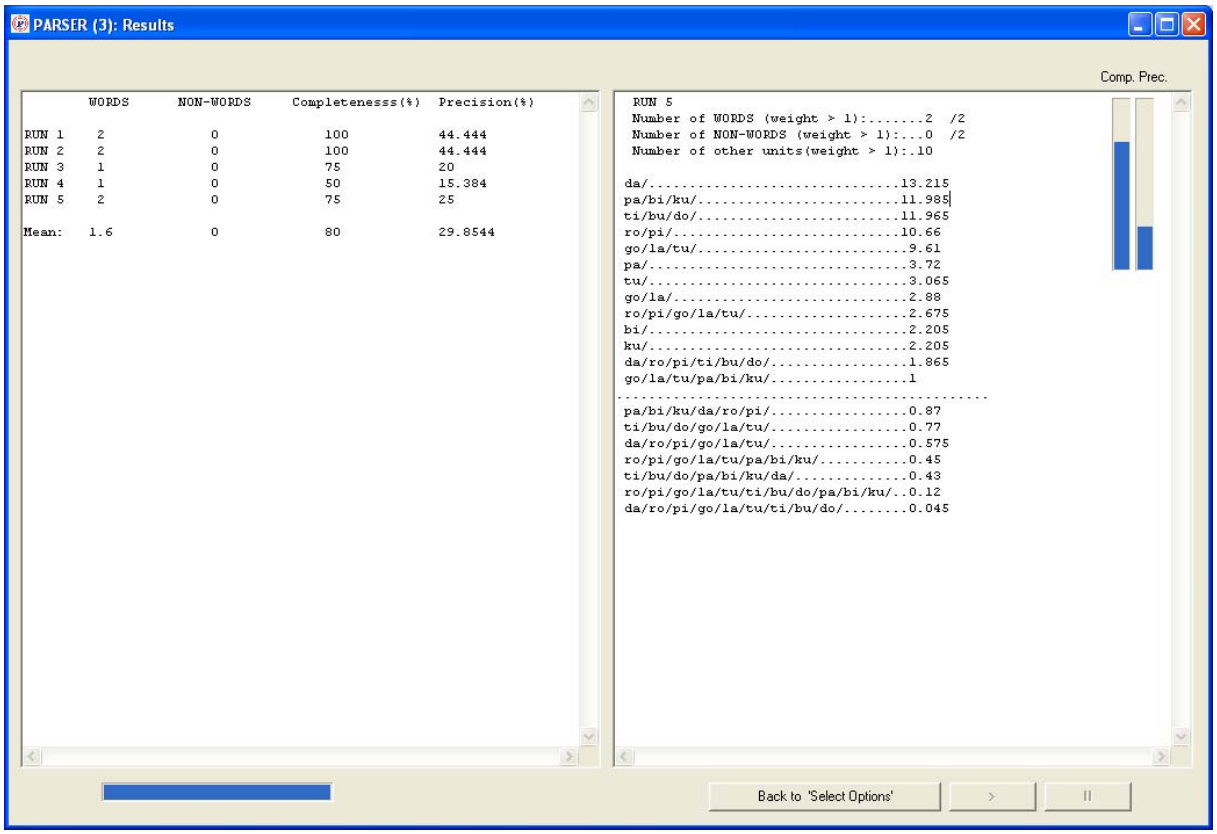
If the option *'save the results*' has been selected, the content of this form will be copied at the top of the results file. Note that it is also possible to save the corpus that the program has generated when the '*Generate a corpus'* option has been selected. The corpus(es) will be saved in a file the name of which is identical to the results file, with the suffix '_corpus'.

The program may fail to generate the corpus. It is not possible, for instance, to generate a language without immediate repetition comprising four words, *a*, *b*, *c*, and *d*, the frequency of which is *a*=10, *b*=10, *c*=10, and *d*=100 (or any value > 30). Indeed, the frequency of *a*, *b*, and *c*, is too low to avoid the repetition of *d*. Of course, the program does not assess the intractability of the problem through analytical means: it all simply gives up after 100.000 unsuccessful iterations. In this case, a message: "'*Parser fails to built a corpus. Please change the parameters*" appears.

3 - Analyze the results

The result window comprises two main frames. Under the *'Step-by-step'* mode, the right-hand frame displays the result of the current step, and the left-hand frame displays the results for Step N-1, hence allowing analysis of the operations performed by the model on each step.

Under the '*Normal'* mode, the right-hand frame displays the final state of the current run, and the left-hand panel displays a record of the final scores for each run. The content of these frames is reported on the result file (if the option *'Save the results*' has been selected), with the content of the right-hand frame being recorded in succession for each run (if several runs have been required), and the content of the last left-hand frame being appended to the file.



The two bars in the high-right corner indicate the scores of completeness (the proportion of words that are extracted) and precision (the proportion of actual words among the extracted units), respectively. Note that the scores of completeness and precision are correct only if all the words (and only the words) of the language have been previously provided. This is obviously the case under the '*Generate a corpus*' option, given that the corpus is created on this basis, but under the *Load* mode, the program has no means to check that the words have been correctly entered.

If test lists have been provided, the program also returns the number of discovered items belonging to each list (e.g., test words and test part-words), and the number of items that have been found but which do not belong to the list(s).

Appendix: Source of the 'Ready to use examples', with a few comments...

Aslin, R. N., Saffran, J. R., & Newport, E. L. (1998). Computation of conditional probability statistics by 8-month-old infants. *Psychological Science,* 9, 321-324.

The first study using a 'frequency-balanced design'. In a 'frequency-balanced' design, some items are more frequent than other items in the familiarization speech. This allows to have test words and test part-words of equal frequency, but differing with regard to the transitional probability between their constituents. Note that the numbers of items that are displayed are those used in Aslin et al. In fact, the correct values to obtain a genuine frequency-balanced design would be 47 and 88, instead of 45 and 90 (French & Perruchet, BRM 2009).

Giroux, I., & Rey, A. (2009). Lexical and sub-lexical units in speech perception. *Cognitive Science,* 33, 260-272.

This study compares the recognition performance of adults for lexical and sublexical units of same length after hearing 2 or 10 min of an artificial spoken language. The results are consistent with Parser's predictions, showing that performance on words is better than performance on part-words only after 10 min. Note that simulating all the results involve several changes in the provided material: (1) The frequency of words (145) is for 10 min of exposure. For 2 min, the value needs to be set to 29. (2) The test items are words and sublexical units, not part-words (see the Appendix A in the paper for details).

Perruchet, P., & Desaulty, S. (2008). A role for backward transitional probabilities in word segmentation? *Memory & Cognition,* 36, 1299-1305.

This study shows that adult participants are sensitive to the standard, 'forward', transitional probabilities, but also, more surprisingly, to backward transitional probabilities. Parser predicted this result, while a SRN is unable to account for it. The provided material is the one used in Experiment 2, in which the raw frequency is controlled, as in Aslin et al, 1998.

Perruchet, P. & Peereman, R. (2004). The exploitation of distributional information in syllable processing. *Journal of Neurolinguistics,* 17, 97-119

The paper reports an experiment collecting judgments of word-likeness as a function of the relationship between the phonemes composing the rimes (VC) of monosyllabic words. The contingency between Vs and Cs, as assessed by rphi (the normative measure of contingency) was the best predictor of children and adult judgments, and the backward transitional probability

(pV/C) made a sizeable contribution. Parser proved to be a better predictor of performance than an SRN (but better results are obtained if the role of interference in forgetting is increased –e.g. decay=0.025 and interference=0.025).

Perruchet, P., & Vinter, A. (1998). PARSER : A model for word segmentation. *Journal of Memory and Language*, 39, 246-263.

The provided material is the one used in Study 4. Parser turns out to be able to discover a word (' bu') that is a component of larger words (e.g., 'dutabu').

Saffran, J. R., Aslin, R. N., & Newport, E. L. (1996). Statistical learning by 8-month-old infants. *Science*, 274, 1926-1928.

One of the two seminal papers that prompted research on statistical learning.