The StaGe Software Package (SSP) - manual

Frans Adriaans

March 22, 2010

Abstract

This manual describes how to use the StaGe Software Package (SSP). The software can be used to run phonotactic learning and speech segmentation simulations, as described in Adriaans and Kager (2010, *Journal of Memory and Language*). In addition, it allows the user to train models on new data sets, with the possibility to use other statistical measures, thresholds, and a different inventory of phonological segments and features. The package implements several learning models (StaGe, transitional probability, observed/expected ratio) and segmentation models (Optimality Theory, threshold-based segmentation, trough-based segmentation). Please see Adriaans and Kager (2010) for details about these models. This manual describes how to run simulations with SSP.

1 Getting started

The software (`SSP.zip`) can be downloaded from:

`http://www.let.uu.nl/~Frans.Adriaans/personal/resources/`

The program uses configuration files, which specify how to run the software. In addition, the user will need to provide a training/test corpus and an inventory file specifying phonological segments and features[1]. The contents and required format of these files are described in the next sections.

1.1 Contents of the package

- `ssp.pl` - The main script, written in Perl.
- `Input` - This is the folder where the corpus and inventory files should be placed.
- `Modules` - This contains several Perl modules which are used by the main script.
- `Output` - This is the folder where the program writes its output to. A new folder will be created for each configuration file that is used to run the program.
- `ssp_manual.pdf` - This manual.
- Several configuration files (`.txt`) have been included which can be used to replicate the simulations of Adriaans and Kager (2010), and which can be modified to create new simulations.

[1] Examples of configuration files and an inventory file for Dutch are included in the package. The Spoken Dutch Corpus (*Corpus Gesproken Nederlands*) is distributed by the Dutch HLT Agency (TST-centrale; `http://www.inl.nl/tst-centrale`) and can be ordered there.

1.2 Running the program

To start the program, use Command Prompt (Windows) or Terminal (Unix, Mac OS X) to go to the SSP folder and type:

```
perl ssp.pl configuration.txt
```

where the configuration file 'configuration.txt' can be replaced by any other file name. Please note that running Single-Feature Abstraction may take several minutes, while the segmentation of *wxyz* sequences ('segmentation using context') may take up to several hours, depending on the speed of your computer.

2 Input files

All input files (corpus files, feature definitions) should be placed in the Input folder.

2.1 Corpora

Input to the program consists of the following:

- **Training file** - A training file is a single text (.txt) file with phonemic transcriptions. Each segment should be represented by a single ASCII character (such as in the CELEX DISC alphabet). Each line contains a sequence of segments. This can either be a single word or multiple words glued together. The sequences are optionally preceded by a digit (e.g. utterance IDs) and optionally followed by a digit specifying a (token) frequency count. The entries should be tab-separated.

  For example:

  (a) A corpus of transcribed utterances of continuous speech (with utterance IDs):

  ```
  5    dizKnv@rstElbardor@tsKnsxufpot@
  6    d}s@hKstadnyOps@nlaxlaxst@stAnt
  7    laG@rdAnsokAni
  ...  ...
  ```

  (b) A lexicon of isolated words (with token frequencies):

  ```
  dQg  2061
  k{t  1195
  ...  ...
  ```

  (c) Sequences of segments:

  ```
  mademobetumopodemopotubipotumo
  ...
  ```

- **Test file** - The trained model can be used to predict the locations of word boundaries in a test set. Similar to the training file, a test file consists of sequences of segments, optionally preceded by a digit to identify the utterance.

- **Correct file** - A 'correct' file can be specified to evaluate the model's segmentation of the test set. In order to link the correct segmentation of an utterance to the predicted segmentation

of the test set, the user can specify utterance ID digits in both files, so that the program knows which correct utterance corresponds to which predicted utterance in the test set. For example:

```
6  d} s@ hKstad ny Op s@n lax lax st@stAnt (Test file)
6  d}s @ hK stad ny Op s@n lax laxst@ stAnt (Correct file)
```

If no utterance IDs have been specified, the program will assume identical orderings in both files (i.e., the first utterance in the test file corresponds to the first utterance in the correct file, etc.).

2.2 Segment/feature inventory

A tab-separated text file is used to define the segment inventory with corresponding feature values. Each line contains one such specification:

```
Segment1  Feature1=Value1  Feature2=Value2  ...
Segment2  Feature1=Value1  Feature2=Value2  ...
   ...          ...              ...         ...
```

For example:

```
 p   ...  Cons=1  Voice=0  Place=Lab  ...
 b   ...  Cons=1  Voice=1  Place=Lab  ...
...  ...   ...      ...       ...     ...
```

Note that such an inventory only needs to be specified when using StaGe. The statistical learning models (TP, O/E) do not require inventory files. The user is free to use different names for features and feature values (as long as they are consistent within the file). There is no restriction with respect to possible feature values[2]. That is, any two different values for a single feature is counted as a feature difference (e.g., `Voice=0` $\neq$ `Voice=1`; `Voice=+` $\neq$ `Voice=-`, `Voice=nice` $\neq$ `Voice=notsonice`, etc.).

Some restrictions hold with respect to feature definitions. First, the order in which the features are specified for each segment should be fixed. Second, the file should not contain multiple segments with identical feature specifications. We suggest that segments with such zero-feature differences be replaced by a single (segment) symbol representing that feature specification. Third, the model only processes constraint pairs that have feature vectors of the same length. Each segment should thus be fully specified. Note that the user may use different features for consonants and vowels. In this case, consonant and vowel vectors will have different lengths and are not compared to each other. The consequence is that the model only compares CV constraints to CV constraints, CC constraints to CC constraints, etc. An example of such a feature definition is included in the software package (`inventory-dutch.txt`). Finally, please make sure that all segments that occur in the corpus are defined in the inventory.

3 Configuration files

A configuration file is a text file specifying how to run the software. The user may modify parameters in configuration files in order to apply models to new data sets, or may alter the way in which a

[2]With the exception of the symbol '?', which is used for program-internal purposes

model processes its input, when it induces constraints, etc.[3] (See also the example configuration files that are included in the package.) Table 1 shows some basic parameters for the program, such as a specification of which learning model is to be used (`MODEL`); which training and/or test data the model should be applied to (`TRAININGSET`, `TESTSET`); where to find the the correct segmentations for the test set (`CORRECT`); and, if StaGe is used, which feature set should be used (`INVENTORY`).

Several input processing parameters are given in Table 2. The user controls what type of sequences are extracted from the corpus (`TRAININGPATTERN`). The default sequence is a simple biphone pattern (`TRAININGPATTERN = xy`). The software also supports more sophisticated processing windows, such as non-adjacent dependencies. For example, the model can be trained on non-adjacent consonants by specifying a larger, feature-based pattern (`TRAININGPATTERN = [syll=0][syll=1][syll=0]`[4]). When such a larger window is used for training, the user needs to specify the positions for which the statistical dependency is to be computed. In the current example, a dependency between the first and third positions needs to be specified (i.e., ignoring the vowel in the second position): `DEPENDENCY = 1-3`. The user may specify whether token frequencies (which are optionally included in the training set) are to be used in training the model (`COUNT`).

It should be noted that the resulting model is always 'biphone'-based. That is, only the two dependent elements are encoded in the model (e.g., *CC rather than *C(V)C). The constraints are applied directly to the test set (i.e., the test set is not pre-processed in any way). Therefore, when non-adjacent constraints are used, the user may want to filter the test set accordingly (e.g., removing all vowels from a test set containing CVCVCV... sequences, resulting in a test set containing only CCC... sequences). During testing the model either uses context or not (`CONTEXT`). That is, the test set is processed using either an *xy* (i.e., `CONTEXT = no`) or a *wxyz* (i.e., `CONTEXT = yes`) window.

Several additional (optional) parameters control more technical aspects of the learning and segmentation models (see Tables 3 and 4). The user can change the statistical learning formula that is used to create the statistical distribution (`SLFORMULA`) using any of the measures described in Table 5; change the statistical measure that is used to compute constraint ranking values (`RANKING`); change the constraint induction thresholds (`THRESHOLD_M`, `THRESHOLD_C`) for StaGe; and change the segmentation thresholds (`THRESHOLD_OE`, `THRESHOLD_TP`) for the statistical learning models. Finally, a default boundary probability (`BOUNDARYPROB`) can be specified for missing biphones, random baselines, etc.

4 Output files

For each simulation, a new folder will be created within the `Output` folder. This folder carries the name of the configuration file that was used in the simulation and, depending on the particular configuration that was used, will contain one or more of the following output files:

- *Configuration*`-model.txt`
  - ...contains either the constraint set with ranking values (StaGe), or biphones with associated transitional probabilities or observed/expected ratios (TP, O/E).

[3] The flexibility provided by the various parameters in SSP is provided to encourage and facilitate research on phonotactic learning and speech segmentation. It is the responsibility of the user, of course, to decide whether a particular new configuration makes any theoretical/empirical sense.

[4] The features that are used in the pattern are defined by the user in the segment/feature inventory. A pattern may also involve multiple (comma-separated) features (e.g., `[syll=0, son=0, cont=1]`)

- *Configuration*-`decisions.txt`
  - ... contains all sequences that were processed in the test set (i.e., either xy or $wxyz$ sequences). For each sequence classification is given, which is the probability of a boundary appearing in the middle of the sequence. That is, a classification of '0' means that a boundary is never inserted, the classification '1' means that a boundary is always inserted, and values between 0 and 1 indicate the probability of the insertion of a boundary into sequence in the test set. The final column displays how often the sequence appears in the test set.
- *Configuration*-`segmentation.txt`
  - ... contains the segmentation of the complete test set, as predicted by the model.
- *Configuration*-`results.data`
  - ... contains evaluation metrics (hit rate, false alarm rate) reflecting the model's ability to predict word boundaries in the test set. More precisely:

Predicted segmentation	**Correct segmentation**	**Label**
'$x.y$'	'$x.y$'	$TruePositive$
'$x.y$'	'xy'	$FalsePositive$
'xy'	'xy'	$TrueNegative$
'xy'	'$x.y$'	$FalseNegative$

Hit rate (H):

$$H = \frac{TruePositives}{TruePositives + FalseNegatives} \tag{1}$$

False alarm rate (F):

$$F = \frac{FalsePositives}{FalsePositives + TrueNegatives} \tag{2}$$

- *Configuration*-`tableaus.txt`
  - ... contains OT tableaus for each sequence in the test set. The optimal candidate is indicated by an arrow '->' and decides the classification (boundary probability) of the sequence. *(only applies to* StaGe*)*

Appendix - Configuration file parameters

Table 1: Basic model and file parameters. ([1] = required fields, [2] = required if StaGe is used.)

Parameter	Possible values	Description
MODEL[1]	StaGe, O/E, TP, random	The learning model
TRAININGSET[1]	File name	Training file
TESTSET	File name	Test file
CORRECT	File name	Correct file
INVENTORY[2]	File name	Segment and feature definitions

Table 2: Input processing parameters.

Parameter	Possible values	Description	Default
TRAININGPATTERN	xy [Feat1=Val1][Feat2=Val2][etc.]	Phonological pattern used for training.	xy
DEPENDENCY	1-3, 1-2, 2-3, etc.	The positions of the elements for which the statistical dependency is to be computed.	-
COUNT	type, token	Specifies whether word frequency counts in the training file should be taken into account.	type
CONTEXT	yes, no	Specifies whether context is to be used during segmentation.	no

Table 3: Learning parameters for StaGe.

Parameter	Possible values	Description	Default
SLFORMULA	(See Table 5)	Formula that is used to implement statistical learning.	O/E
RANKING	(See Table 5)	Statistical measure that is used to rank the constraints.	E
THRESHOLD_M	Any number.	Statistical threshold for the induction of a markedness constraint.	0.5
THRESHOLD_C	Any number.	Statistical threshold for the induction of a contiguity constraint.	2.0

Table 4: Segmentation parameters for statistical learning models (TP, O/E) and random baselines (random). ([1]If no fixed TP threshold is specified by the user, the program will calculate a threshold based on the default assumption that all successors of a segment are *a priori* equally probable. The threshold thus equals $\frac{1}{|Y_x|}$, where $|Y_x|$ is the number of possible successors for x.)

Parameter	Possible values	Description	Default
THRESHOLD_OE	Any positive real number	Segmentation threshold for O/E threshold-based segmentation.	1.0
THRESHOLD_TP[1]	Any probability between 0 and 1	Segmentation threshold for TP threshold-based segmentation.	$Threshold_{xy} = \frac{1}{\|Y_x\|}$
DEFAULTPROB	Any probability between 0 and 1	Probability of inserting a word boundary in case the learning model cannot decide (e.g., unseen biphones, random baselines)	0.5

Table 5: Statistical measures.

Symbol	Description
O	observed frequency (= biphone frequency of occurrence)
Fx	observed frequency of "x_" (= frequency of first-position uniphone)
Fy	observed frequency of "_y" (= frequency of second-position uniphone)
E	expected frequency (= uniphone-based estimated biphone frequency)
O/E	observed / expected ratio
O-E	observed minus expected (= the difference between O and E)
logO/E	log observed/expected ratio, with base 10
MI	pointwise mutual information (= log O/E ratio, with base 2)
TP	forward transitional probability
logTP	log forward transitional probability, with base 10
TPb	backward transitional probability
logTPb	log backward transitional probability, with base 10