FAAL .jar library 1.0 – Introduction and basic implementation

July 2, 2018

FAAL is distributed as a java *.jar* library.

To use it, just import the *.jar* library and copy the *config* folder into the main folder of the java project.

The whole algorithm is implemented in the method *FAAL.faal()*. Independent methods for the IPA parser and for the Matcher modules are also present (see below).

The easiest way to use FAAL is just to run the *FAAL.faal()* method with the two words to be aligned as arguments. The syntax in this case would be:

List<List<Object>> = FAAL.faal(Word_A, Words_B)

In this way the algorithm is run with the default settings (the same used in the test case described in the article). No special tuning or customization is needed.

By default, the algorithm recognizes both IPA characters and diacritics. Both diacritics modifying the preceding character and diacritics modifying the following one are allowed (see below). The full list of characters and diacritics is available in the configuration files *config/phon_feature.txt* and *config/phon_diacritics.txt*. Additional characters, associated with a specific set of phonetic features, can be added by modifying those files.

A few rules have to be followed in the transcription of the words. In particular:

1. Morphemic boundaries must be marked with the character ¦(U+FFE4).
2. Diphthongs must explicitly marked.
   Rising diphthongs must be coded with the diacritic ◡ (U+035C) after the first element of the diphthong (e.g. ɪa͡ = ɪ + ◡ + a), while falling diphthongs must be coded with the diacritic ◠ (U+032F) after the second element of the diphthong (e.g. aɪ̯ = a + ɪ + ◠).
   The diacritics for rising and falling diphthongs can be combined to transcribe triphthongs.
3. Coarticulated phonemes are marked with the diacritic ◠ (U+0361) placed between the two phonemes (e.g. p͡f = p + ◠ + f). The features of the resulting coarticulated phoneme will correspond to the combination of the features of the two original phonemes. Where their features are different, both options will be taken into account.

For a detailed description of the format of the output, see below. Suffice it here to say that in the resulting alignments (see *FAAL.faal*, output, 0. and 1. below) the "0" is used to mark a gap in the alignment, and the sign ¦(U+250A) is used to indicate the first and last pair of the alignment (and therefore it can be used to recognize possible suffixes or prefixes). In addition, in the resulting alignment without diacritics (see *FAAL.faal*, output, 2. and 3. below) the : is

used to divide the various phonemes while ; is used to mark phonemes that have not been counted in the calculation of the Similarity Scores (see below, some settings allow the exclusion of some phonemes from the calculation).

FAAL – detailed description

More complex customized configurations are also possible as alternatives to the basic implementation just described.

In particular, various options can be set for the *FAAL.faal()* method described above, while two other methods, corresponding to the *IPA_parser* and to the *Matcher* modules described in the article, are also present in the FAAL package. All three methods can either be used in their basic form with default parameters, or can be customized.

They are described in detail below.

Name: FAAL.faal

- Description:

  Method containing the whole algorithm. Various settings can be configured. It parses one pair of words at a time, and it returns an array list of possible alignments organized according to their Global or Corrected Global Similarity Score.

- Syntax – with default values:

  List<List<Object>> = FAAL.faal(Word_A, Words_B)

- Syntax – customizable:

  List<List<Object>> = FAAL.faal(Word_A, Words_B, IPA_configs, Matcher_configs_1, Matcher_configs_2, Matcher_configs_3, Matcher_configs_4)

- Input – Arguments:

0. Word_A (String): IPA transcription of the first word to be aligned.
1. Word_B (String): IPA transcription of the second word to be aligned.
2. IPA_configs (List<String>): configuration settings - see below.
3. Matcher_configs_1 (List<Boolean>): configuration settings – see below.
4. Matcher_configs_2 (List<Integer>): configuration settings – see below.

5. Matcher_configs_3 (List<String>): configuration settings – see below.
6. Matcher_configs_4 (List<Double>): configuration settings – see below.

- Output – Results:

  List<List<Object>>

  Each *List<Object>* item within the List corresponds to an alignment. They are organized according to their Global or Corrected Global Similarity score (see below *config files*). Each of these *List<Object>* item contains:

0. *String*: Alignment for Word A
1. *String*: Alignment for Word B
2. *String*: Alignment for Word A without diacritics
3. *String*: Alignment for Word B without diacritics
4. *Double*: Global Similarity Score
5. *Double*: Corrected Global Similarity Score
6. *List<String>*: List of phonetic pairs attested within the alignment. Each item on the list corresponds to a phonetic pair, and it is stored as a string with the following syntax: "phoneme_A - phoneme_B"
7. *List<Integer>*: Number of attestations within the alignment for each phonetic pair of 6. here above.

Name: IPA_parser.IPA_parser_new

- Description:

  Method corresponding to the *IPA_parser* module. It parses one pair of words at a time. Various settings can be configured (see below). It returns a transcription of the words without diacritics, a basic feature matrix and a matrix corrected according to the salience settings.

- Syntax – with default values:

  List<Object> = IPA_parser.IPA_parser_new(Word_A, Word_B)

- Syntax – customizable:

  List<Object> = IPA_parser.IPA_parser_new(Word_A, Word_B, IPA_configs)

- Input – Arguments:

    0. Word_A (String): IPA transcription of the first word to be aligned.
    1. Word_B (String): IPA transcription of the second word to be aligned.
    2. IPA_configs (List<String>): configuration settings - see below

- Output – Results:

    List<Object>

    Such *List<Object>* contains:

    0. *String*: Parsed (without diacritics etc.) IPA transcription of word A
    1. *String*: Parsed (without diacritics etc.) IPA transcription of word B
    2. *int[][]*: feature matrix corrected according to the salience settings. The dimensions of the matrix correspond to the length of the parsed (i.e. without diacritics etc.) transcription of word A × the length of the parsed (i.e. without diacritics etc.) transcription of word B
    3. *int[][]*: basic feature matrix without any salience correction. Dimension as previous one.

Name: Matcher.Matcher_new

- Description:

    Method corresponding to the Matcher module. It parses one pair of words at a time. Various settings can be configured. It returns an array list of possible alignments organized according to their Global or Corrected Global Similarity Score.

- Syntax – with default values:

    List<List<Object>> = Matcher.Matcher_new(Parsed_Word_A, Parsed_Word_B, comp_matrix, comp_matrix_orig, Word_A, Word_B)

- Syntax – customizable:

    List<List<Object>> = Matcher.Matcher_new(Parsed_Word_A, Parsed_Word_B, Comp_Matrix, Comp_Matrix_Orig, Word_A, Word_B, Matcher_configs_1, Matcher_configs_2, Matcher_configs_3, Matcher_configs_4)

- Input – Arguments:

    0. Parsed_Word_A (String): parsed (i.e. without diacritics etc) transcription of word A (returned by IPA_parser.IPA_parser_new - output 0. above)
    1. Parsed_Word_B (String): parsed (i.e. without diacritics etc) transcription of word B (returned by IPA_parser.IPA_parser_new - output 1. above)
    2. Comp_Matrix (int[][]): matching features matrix corrected according to the salience settings (returned by IPA_parser.IPA_parser_new - output 2. above).
    3. Comp_Matrix_Orig (int[][]): basic matching features matrix without any salience corrections (returned by IPA_parser.IPA_parser_new - output 3. above).
    4. Word_A (String): IPA transcription of the first word to be aligned.
    5. Word_B (String): IPA transcription of the second word to be aligned.
    6. Matcher_configs_1 (List<Boolean>): configuration settings – see below
    7. Matcher_configs_2 (List<Integer>): configuration settings – see below
    8. Matcher_configs_3 (List<String>): configuration settings – see below
    9. Matcher_configs_4 (List<Double>): configuration settings – see below

- Output – Results:

    List<List<Object>>

  Same output as the *FAAL.faal* method above.

FAAL – Configuration

The customizable versions of the methods above can be configured both by setting some variable in the algorithm and through some configuration files. The variables can be set to modify specific aspects of the algorithm. They are collected into Array Lists and set as arguments of the methods (see above). The configuration files, instead, are used to define the set of features and modifiers, the salience settings, and some properties associated with specific phonemes.

FAAL - Configuration - variables

IPA_configs (List<String>)

- DESCRIPTION:

  String array list with variables for the configuration of the *FAAL.faal* and *IPA_parser.IPA_parser_new* modules. The variables are the following:

  0. name of the configuration file storing the list of phonetic features (see below);
     – default value: "config/phon_features.txt"
  1. name of the configuration file storing the list of diacritics modifying the phonetic features;
     – default value: "config/phon_diacritics.txt"
  2. name of the configuration file storing the salience settings for consonants (see below);
     – default value: "config/salience_features_cons.txt"
  3. name of the configuration file storing the salience settings for vowels (see below);
     – default value: "config/salience_features_vows.txt"
  4. name of the configuration file storing the salience settings for semiconsonants (see below);
     – default value: "config/salience_features_semi.txt"
  5. option to parse both consonants and vowels, or only consonants. Options:
     – "yes_vow" : both vowels and consonants are parsed and aligned.
     – "no_vow" : only consonants are parsed; vowels are ignored.

     – default value: "yes_vow"

Matcher_configs_1 (List<Integer>)

- DESCRIPTION:

  Boolean array list with variables for the configuration of the *FAAL.faal* and *Matcher.Matcher_new* modules. All of these options have only two possible values, namely "true" or "false". The variables are the following:

  0. Print data about the alignments while running the algorithm;
     – default value: true
  1. Introducing a minimum limit of features in common for a pair to be counted in the calculation of the Similarity Scores. If the number of features is above such a limit, they are counted. If it is below, they are not. The value of such a limit is set in matcher_configs_2. The option here determines whether the limit is taken into consideration

or not. If set to false, it is ignored;
– default value: false

2. Automatically detecting the presence of morphemic boundaries and adapting the Similarity Score used;
– default value: true
3. If "semiconsonats"[1] do not match with anything, they are not counted in the calculation of the Similarity Scores. This option was conceived to reflect the fact that some phonemes (usually semiconsonants) can disappear or be assimilated more easily than other consonants. Note that every phoneme can be labelled as a "semiconsonant" through the "config/phon_features.txt" configuration file;
– default value: false
4. If vowels (marked as such in the "config/phon_features.txt" configuration file) do not match with anything, they are not counted in the calculation of the Similarity Scores. This option was conceived to reflect the fact that vowels are less stable than other phonemes;
– default value: false
5. If vowels (marked as such in the "config/phon_features.txt" configuration file) do not match with anything, they are not counted in the calculation of the Similarity Scores unless they are located at the beginning of the word. If this option is true, the previous one is ignored;
– default value: false
6. Display in the result the limits of the aligned pairs by marking them with ¦;
– default value: true
7. Match only phonemes belonging to the same category, as defined in the "config/phon_features.txt" configuration file;
– default value: true

Matcher_configs_2 (List<Boolean>)

- DESCRIPTION:

  Integer array list with variables for the configuration of the *FAAL.faal* and *Matcher.Matcher_new* modules. The variables are the following:

  0. Select the Similarity Score to be used: – Options:

[1] Here the term semiconsonants is used in a conventional and somehow loose way. It does not refer only to true semiconsonants, but is rather used to refer to any phonemes that could tend to disappear or be assimilated and which could therefore be less relevant in the alignment process. Semiconsonants and glides are the best examples of that, and they are marked accordingly by default in the FAAL package. These settings, however, can be changed, and other phonemes could be marked in this way.

– 0 : use the Global Similarity Score and organize results accordingly.
– 1 : use the Corrected Global Similarity Score and organize results accordingly

– Default value: 0

1. Set the minimum number of features that two consonants must have in common to be counted in the calculation of the Similarity Scores. This limit will be taken into consideration only if the option 1 in the *Matcher_ Configs_ 1 (List<boolean>)* is set to "true" (see above). Setting the value to 0 allows for any pair to be taken into consideration:
– Options: any number between 0 and the total number of features listed in the "config/phon_features.txt" configuration file;
– Default value: 23

2. Set the minimum number of features that two vowels must have in common to be counted in the calculation of the Similarity Scores. This limit will be taken into consideration only if Option 1 in the *Matcher_ Configs_ 1 (List<boolean>)* is set on "true" (see above). Setting the value to 0 allows for any pair to be taken into consideration:
– Options: any number between 0 and the total of features listed in the "config/phon_features.txt" configuration file;
– Default value: 23

3. Set the number of pairs with the highest scores that the algorithm will take into consideration to build the possible alignments (see Matcher 3. in the article), while morphemic boundaries are present:
– Options: any number > 0
– Default value: 6

4. Set number of pairs with the highest scores that the algorithm will take into consideration to build the possible alignments (see Matcher 3. in the article), while no morphemic boundary is present: – Options: any number > 0
– Default value: 6

Matcher_configs_3 (List<String>)

- Description:

String array list with variables for the configuration of the *FAAL.faal* and *Matcher.Matcher_new* modules. The variables are the following:

0. name of the configuration file storing the salience settings for consonants (see below). It must be the same as IPA_configs (List<String>), 2; – default value: "config/salience_features_cons.txt"
1. name of the configuration file storing the category attributed to each phoneme (see below); – default value: "config/cons_vows_semi.txt"
2. name of the configuration file storing the list of diacritics modifying the phonetic features (see below); – default value: "config/phon_diacritics.txt"

Matcher_configs_4 (List<Double>)

- Description:

  Double array list storing the factors used in the calculation of the Corrected Global Similarity score. It is used in the configuration of the *FAAL.faal* and *Matcher.Matcher_new* modules:

0. Factor A – approximate average of the salience for manner and place features as defined in the default settings; – default value: 7.71
1. Factor B – obtained empirically; – default value: 2
2. Factor C – general scaling factor (see below); – default value: 3
3. Factor D – general scaling factor (see below); – default value: 8

FAAL – Configuration – configuration files

config/phon_features.txt

- Description:

  File containing the definition of the phonetic features taken into consideration for each phoneme. The phonetic features and their values used by default are taken from Hayes (2009:95–7).

- Format:

  Lines starting with % are ignored.
  Each phoneme occupies a different line, and each feature a different column. Features can have 4 different statuses, namely: [+]; [-]; [±]; [0]. The status [+] can match with itself and with [±]. The same stands true for [-]. The status [0], instead, matches only with itself. The number and nature of features taken into consideration can be modified, although the number of features for each phoneme in the file must be the same. The only exception is the first three features, which correspond to "diphthong" (= first feature), "syllabic" (= second feature) and "consonantal" (= third

feature). These three features must always be present and must be organized in this order. By default, the feature "diphthong" must be set to [0] for each phoneme. Vowels have [+] "syllabic" and [-] "consonantal". Consonants have [-] "syllabic" and [+] "consonantal". "Semiconsonants" (see above *Matcher_ configs_ 1(List<Boolean>)*, 3.), instead, conventionally have [-] "syllabic" and [-] "consonantal".

config/phon_ diacritics.txt

- DESCRIPTION:

  File containing the definition of the phonetic features modified by diacritics. Both diacritics modifying the preceding phoneme and diacritics modifying the following one are allowed.

- FORMAT:

  Lines starting with % are ignored.
  Each diacritic occupies one line. The first parameter corresponds to the diacritic. The second parameter defines whether the diacritic modifies the preceding phoneme (in this case its value is set to 0) or the following one (in this case it is set to 1). The third parameter indicates which feature of the preceding or following phoneme is modified, and how (e.g. 11;+ means that the feature 11 is turned into [+]). More than one feature can be modified at the same time by the same diacritic.

config/salience_ features_ cons.txt

- DESCRIPTION:

  File containing the salience settings for consonants.

- FORMAT:

  Lines starting with % are ignored.
  Each feature occupies a line. The first parameter corresponds to the number of the feature in the *config/phon_ features.txt* file, the second to its salience.

config/salience_ features_ vows.txt

- DESCRIPTION:

  File containing the salience settings for vowels.

- Format:

  Lines starting with % are ignored.
  Each feature occupies one line. The first parameter corresponds to the number of the feature in the *config/phon_features.txt* file, the second to its salience.

config/salience_features_semi.txt

- Description:

  File containing the salience settings for "semiconsonants".

- Format:

  Lines starting with % are ignored.
  Each feature occupies one line. The first parameter corresponds to the number of the feature in the *config/phon_features.txt* file, the second to its salience.

config/cons_vows_semi.txt

- Description:

  File containing the classification of phonemes into consonants, vowels, and "semiconsonants".

- Format:

  Lines starting with % are ignored.
  Each phoneme occupies a line. The first parameter corresponds to phoneme, the second to its classification. Consonants are marked with 0, "semiconsonants" with 1, and vowels with 2.

FAAL – Configuration – configuration files

The functions to calculate the Global and Correct Global Similarity Scores used by default are those presented in the article. Their results have been divided by a general scaling factor (of 8 and 3 respectively) to reduce them to a comparable order of magnitude.
The two functions have been implemented as methods and have been stored into two independent files within the *.jar* library (*glob_sim_score.java* and *corr_glob_sim_score.java* respectively) so that the user can easily modify or replace them altogether with customized ones.