

SYLVIANE CARDEY
RICHARD SCHMIDT
Centre Tesnière, University of Bourgogne Franche-Comte
Besançon, France
sylviane.cardey@univ-fcomte.fr, richard.schmidt@edu.univ-fcomte.fr

Recognising acronyms in technical documentation and in general language¹

1. Introduction

Acronyms can display linguistic phenomena which can present serious problems involving ambiguity, for example confusion with abbreviations, synonymy and homonymy (including with normal lexis). In this paper we present a technique that we have developed for the automatic recognition of acronyms including their associated full forms together with their appearances in a predefined contextual sense dependent situation for technical documentation, and in particular that used in safety critical applications². This technique provides the information necessary to draw to the attention of the documentation control personnel potential acronyms that have been recognised and which require their particular consideration. The safety critical nature of the use of such documentation necessitates the provision of both static and dynamic traceability and this too is covered by our technique. The extension of the methodology that we have developed is then shown by means of a formal model for the recognition of acronyms in general for the English language.

The paper is organised as follows. We first give a brief overview of acronyms as a linguistic phenomenon. Following this, the paper is organised in two principal parts: recognising acronyms in technical documentation and recognising acronyms in general language. In the first part we define precisely what an acronym is in respect of its use in the technical documentation in question and we describe the specific linguistic criteria that we have adopted. We then introduce and situate the document processing algorithm that we have developed, describing as necessary the linguistic programming facilities and the traceability facilities that are provided. In the second part, recognising acronyms in general language, a formal model for dealing with general, common English acronym recognition is presented, and this is followed by our conclusion.

2. Acronyms – a linguistic phenomenon

In contemporary texts of all sorts, acronyms abound and they are an increasingly productive phenomenon (Jeandot 2008). Indeed this productivity together with possible confusion between acronyms and ‘normal’ lexis as well as with abbreviations together with the absence of a precise definition as to what is and what is not an acronym means that their introduction, with or without defining full forms, often passes unobserved.

Acronyms such as “XML” for “Extensible Markup Language”, “GABA” for “γ-Aminobutyric acid”, “2MASS” for “Two-Micron All Sky Survey”, “GNU” for “GNU’s not Unix” or “I²L” for “Integrated-injection logic” illustrate the complexity of the phenomenon.

This situation requires to be controlled in technical documentation and particularly so for that used in safety critical applications.

3. Part I: Recognising acronyms in technical documentation

In the case of technical documentation, and in particular such as is used in safety critical applications, uncontrolled acronyms can display linguistic phenomena which can present serious problems. Before we can even attempt to deal with these linguistic phenomena, we need to have a precise definition of what an acronym is. For example, at <http://www.merriam-webster.com/> (27/04/2009), we find for ‘acronym’:

a word (as NATO, radar, or laser) formed from the initial letter or letters of each of the successive parts or major parts of a compound term

However, research of sources has shown that there is no universally agreed precise definition of what an acronym is (Jeandot 2008). Further below we give a definition of acronym for our own purposes in connection with the particular industrial context in which this work has been undertaken.

3.3.1. Problematic

Uncontrolled acronyms can display linguistic phenomena which can present serious problems in the context of technical documentation; for example:

- i. confusion with abbreviations;
- ii. acronyms exhibiting morphological and morpho-syntactic phenomena proper to the host language;
- iii. acronym synonymy and polysemy: for example in the case of polysemy involving confusion with existing and perhaps uncontrolled acronyms appearing in everyday language. This problem is the acronym equivalent to the polysemy problem that can occur between everyday lexis and specialised lexis;
- iv. the presence or absence of full form definitions;
- v. the presence of multiple different full forms; for a given acronym form, such different full forms can be:

¹ Keynote speech in this congress.

² The research described within this paper has been partially funded by the French National Research Agency (L’Agence nationale de la recherche), the ANR, programme: Concepts Systèmes et Outils pour la Sécurité Globale: Projet LiSe (Linguistique et Sécurité), and by the French-Swiss programme Interreg V: Webso+ project.

- a. synonymous
- b. not synonymous
- vi. homophony between pronounceable acronyms and non-acronyms, that is normal lexis;
- vii. the use of certain acronyms in precise semantic contexts;
- viii. the presence of existing (and thus possibly already known and thus possibly controlled) acronyms;
- ix. the case of new acronyms whose form and usage is to be strictly controlled.

As we shall show, certain of these phenomena can be automatically detected in documents and thus aid acronym controlling in a mechanical manner, and this by means of programmable linguistic criteria.

3.2. *Acronyms recognition and normalisation*

For the purposes of the particular industrial context in which this work has been undertaken, certain criteria have been established to which a controlled acronym must conform, other than existing controlled acronyms which must appear in controlled acronym dictionaries, and known acronym exceptions (to the acronym control criteria) which must appear in acronym exception dictionaries. We have for example:

- *Full form composition criterion*: For example an acronym must be formed at least in part from the full form (the compound term) by the single initial capital letter of successive words with initial capital letters (other lexis being thus ‘noise’); for example ‘ABC’ from the full form “Acc Bbb noise Ccc” where “noise” is noise. We note here that the use of underlined bold is for explanatory purposes.

For the industrial setting in which this work has been carried out, we see here restrictions compared with the citation above “...the initial letter or letters...”; we require just *one* initial letter and furthermore that this be a *capital* letter. Existing acronyms that do not conform to this criterion would be entries in the acronym exception dictionaries.

- *Form-based criterion*: for example a potential controlled acronym (in its canonical form) is a string whose length is bounded (minimum and maximum lengths) with an initial capital letter. Other constraints can dictate the composition of the rest of the acronym, for example whether there can be intervening slashes. For example ‘D/E’ with full form “Ddd/Eee” and ‘F/G’ with full form “FffGgg”.
- *Host language based criteria*
 - *Host language morphology and morpho-syntax systems*: a form-based criterion may have to take account of host language influences concerning incorporation of the (canonical form) acronym in the host language’s morphology and morpho-syntax systems. This leads to the possibility of variant forms, such as a substantive plural in -s (lower case) for English as the host language; for example ‘ABCs’ with canonical acronym form ‘ABC’.
 - *Host/other language/acronym confusion criterion*: the issue here is to reduce if not eliminate confusion between acronyms and conventional lexis, including the problem of loan words (other languages). As this host/other language/acronym confusion criterion merits further explanation we explain this in further detail below when we discuss the legal sequences of graphemes in English.

3.3. *The document processing algorithm*

Given the particular industrial context together with the safety critical context of the ultimate use of the documentation under development, the following algorithm has been developed. We wish to stress that this algorithm has been developed for a particular industrial context with particular technical documentation working procedures, standards and resources.

So as to ensure early capture of erroneous data, the control (and thus the application of the algorithm) occurs early in the design of the technical documentation. Furthermore both static and dynamic traceability and the provision of justification data are inherent elements in the design and implementation of the algorithm as is also the case in the human preparation of associated controlling data used by sub-algorithms (as for example the acronym exception dictionaries and the ‘Legal sequences of graphemes in English’ table described below).

3.3.1. *Algorithm and traceability*

As required by the industrial setting, at each step of the algorithm, a dynamic trace is supplied for observation and tracing purposes; the following information is printed:

- Potential_Acronyms: total number followed by each potential acronym on a line with its position and its context in the document’s text, the potential acronyms being printed in the order of their appearance in the document.
- Potential_Acronyms: total number followed by each acronym on a line on its own with its frequency of occurrence in the document, the whole sorted in high > low frequency.

Here the position is that in the document and the context is the text surrounding the potential acronym, the size of the context ‘window’ being a parameter that can be adjusted by the document control personnel.

In the case of full form definitions and the legal sequences of graphemes, further information concerning respectively the full form definition itself and the illegal and hapax sequences of graphemes is printed and this is described below in the relevant steps of the algorithm.

3.3.2. *The algorithm and its steps*

The document processing algorithm is applied to the document to be controlled and is as follows:

Step 1. Extract potential acronyms. This involves extracting strings which conform to the form based criterion. For example:

AA AAs A/A A/A/A A1A A1A1A A1A/A AAAAAA AAAAAAA

Such strings, representing potential acronyms, are passed to the next step of the algorithm.

Step 2. Remove too short or too long potential acronyms. The maximum length is a programmable parameter. Taking the previous example and minimum and maximum lengths of 2 and 6 respectively,

AAAAAAA

is removed, and the remaining potential acronyms are passed to the next step of the algorithm.

Step 3. Filter with dictionaries. Potential acronyms are filtered out if they appear in two specific dictionaries whose respective contents are already controlled within the particular industrial context: 'abbreviations' and 'acronyms'. The 'acronyms' dictionary is maintained in part by the controlled acronyms resulting from manual analysis of the algorithm's outputs. The remaining potential acronyms, after this filtering process, are passed to the next step of the algorithm.

Step 4. Detect full form definitions. Full form definitions take specific forms and furthermore the programmable specification of these enables their detection. Within a document, such full form definitions can occur regrouped in a preliminary table, or they can occur individually in-line in the text proper; which approach to use itself can constitute a document writing standard and thus a document control criterion.

In the case of full form definitions, for observation and tracing purposes, in the case of potential acronyms which are within the length limits of Step 2, further information concerning the full form definition itself is printed such as its type (acro_ff or ff_acro), whether the full form itself contains an acronym, and the full form definition's context.

Consider the following full form definitions:

- Acronym followed by Full Form (acro_ff):
... LM (Lm Mn) ... MN -Mn No- ... NO 'No Op' ... OP "Op Pq" ... PQ, Pq Qr, ... QR Qr Rs ... RS (Rs noise St) ...
- Full Form followed by Acronym (ff_acro):
... Ab Bc (AB) ... Bc Cd -BC- ... Cd De 'CD' ... De Ef "DE" ... Fg Gh FG ... Hj noise Ij (HI) ...

In the above examples the acronym full form definition parentheses can be space(s) or variously (and), _ and -, ' and ', " and "; other parentheses can be tab(s), as in a preliminary table, or newlines separating the acronym from its full form. The particular parentheses pairs are programmable by the document control personnel.

Let us now look at the following two examples:

- i. Acronym followed by full form (acro_ff):
... MFD (Multiple FF noise Definition) ...
- ii. Full form followed by Acronym (ff_acro):
... Minimal FF noise Description (MFD) ...

In both the above examples we note that we can have a full form which itself contains an acronym: FF (with full form "Full Form").

These two examples i. and ii. taken together as a pair also show a case of multiple full form definitions where for the acronym "MFD", the full forms are not synonymous and therefore represent most probably an erroneous situation. In the case of identical full forms, this may indicate the non-respect of standards concerning for example the proper usage of preliminary definitions and in-line definitions. In the case of synonymous full forms, this presents a possibly dangerous redundancy.

Potential acronyms (from Step 3) not having a full form definition are passed to the next step of the algorithm.

Step 5. Legal sequences of graphemes in English. In respect of the technical documentation under study, so as to reduce confusion between acronyms and conventional lexis, it is deemed judicious that controlled acronyms should not contain sequences of graphemes which are legal in the host language, which in the particular industrial context of the work is that of American English. Thus the technique that we describe does not cover confusion with other language lexis.

To this end a tabular technique using a spreadsheet and which is simple to program linguistically has been implemented in which the maximum length of such sequences is 3 (the minimum thus being 2).

An extract of the spreadsheet contents for the legal sequences of graphemes in English for pairs and triples of letters is given in Table 1 with an explanatory legend in Table 2.

Table 1. Legal sequences of graphemes

ACRONYMS_Legal_Sequences_Graphemes					
Start_References					
∅	COMPETENCE				Nov - Dec 2007
M	www.merriam-webster.com				Nov - Dec 2007
...
X	HAPAX				Jan - Feb 2008
End_References					
Maximum_Length				6	
Start_Attestations					
2 Letters	plus ∅	plus A	plus B	...	plus Z
AA	∅	∅	∅	...	∅
AB	STAB	ABATE-D	ABBEY	...	∅
...
AI	CHAIN	NAIAD-M-X	∅	...	BAIZE-M-X
...
ZZ	NOZZLE	PIZZA	∅	...	∅
End_Attestations					

Table 2. Legal sequences of graphemes: legend

Attestation cell content	Meaning
∅	Indicates no attestation
WORD	Indicates by its presence an attestation. Must be capital letters. Must be < or = Legal_Sequences_Graphemes_Maximum_Length
WORD with no suffix	Attested by competence
WORD with a suffix ‘-’ followed by a letter other than X	The letter is a reference. Must appear in the Reference cells
WORD with a suffix ‘-X’	The word is a hapax attested by competence
WORD with a suffix ‘-’ followed by a letter other than X followed by ‘-X’	The letter is a reference and the word is a hapax

The spread-sheet (see Table 1) is divided in 3 parts:

- i. Reference section
- ii. Maximum Length parameter value
- iii. Attestation section

The attestation section is organised so that it has particular cells corresponding to 2 and 3 letter sequences. The interpretation of the contents of a given cell is given by examination of the explanatory legend in Table 2. From Table 2 it will be seen that a particular cell's contents corresponds to one of the following 3 cases:

- i. “∅” indicating ‘no attested English word of less than or equal to 6 letters (Maximum_Length) containing this sequence’
- ii. an attestation with a Reference of a non-hapax English word containing this sequence
- iii. an attestation with or without (in which case competence) a Reference of a hapax English word containing this sequence

Inspection of the extract of the Reference section in Table 1 indicates that a Reference can be variously absent (reminder ∅ indicating linguistic competence), or a dictionary reference(s) (linguistic performance). The X in the Reference section is a reminder that X in the Attestations section is interpreted as a hapax.

Taking for example:

2 Letters	plus ∅	plus A	plus B	...	plus Z
AI	CHAIN	NAIAD-M-X	∅	...	BAIZE-M-X

- The 2 letter sequence AI is attested in English in the word CHAIN and this relies on the linguist's competence.
- The 3 letter sequence AIA is attested in English in the word NAIAD by *www.merriam-webster.com* in the period Nov - Dec 2007 and this attestation is a hapax; the same argument applies to the 3 letter sequence AIZ with hapax attestation BAIZE.
- The 3 letter sequence AIB, whose corresponding cell content is ∅, is not present in any attested English word of less than or equal to 6 letters (Maximum_Length).

The content of the table contains a static trace with the justification of each relevant 2 or 3 letter sequence's content.

Establishing this table is described in detail in Jeandot (2008); here we note the essentials:

- i. Identify suitable on-line dictionaries which are sufficiently complete and give precise definitions, thus enabling choosing between several candidate attestations for a given sequence of 2 or 3 letters, and to either avoid hapaxes or to be confronted with a hapax and thus record this.
- ii. Take into account particular restrictions imposed on the candidate attestations, such as:
 - a. Linguistic restrictions that prohibit for example portmanteaus, compounds, foreign words, proper nouns and expressions (these unbroken - without spaces, hyphens etc.).
 - b. Technical document requirements imposing length restrictions.

In the case of a potential acronym containing either illegal sequence(s) of graphemes (corresponding to letter sequence cells with \emptyset) or sequence(s) of graphemes which is/are hapax, which validate in part its status as an acronym, the illegal/hapax sequence(s) of graphemes are printed with their position in the acronym and with an indication as to whether they are also hapaxes.

In respect of this algorithmic step, potential acronyms containing legal sequences of graphemes are removed and those containing either illegal sequence(s) of graphemes or sequence(s) of graphemes which are hapaxes are passed to the next step of the algorithm.

Step 6. Reinsert any acronym exceptions using the acronym exception dictionaries. This step is necessary to cope for existing acronyms within the particular industrial context which do not conform to controlled acronym criteria.

Step 7. Identify usage context(s) of acronyms. This concerns the appearances of acronyms in a predefined situation and thus which are sense dependent. Here we use sense mining techniques that we have developed elsewhere (Cardey et al. 2006), (Cardey 2013) and which involve context rule languages. For this application we have added a cursor element 'acro' which matches any current potential acronym within a textual window enclosing the acronym and whose size is specified by an 'acronym in context window' parameter. In the absence of the use of "acro" in a rule, then an acronym must be found within the textual window. A match with an acronym indicates its usage in the predefined situation. The 'acronym in context window' parameter and the context rules can be programmed by the document control personnel.

Step 8. Output remaining unsubstantiated potential acronyms. These are those which require the particular attention of the documentation control personnel, notwithstanding the other output from the algorithm's execution.

3.4. *Human decisions*

These involves the decisions by the documentation control personnel concerning not only the remaining unsubstantiated potential acronyms, but the results of each and every algorithm step and in particular the following steps where the mechanical analysis does not cover 'exceptions':

- Step 4: Detect full form definitions: for example 'Irregular' Full Forms such as for example:
... Jkl Mno (JKMN)
- Step 5: Legal sequences of graphemes in English: for example a careful review of hapax occurrences.

3.5. *The document processing algorithm: conclusion*

In order to aid controlling the correct formulation and usage of acronyms within a particular industrial context, we have developed and implemented a technique for variously the automatic recognition of acronyms, including their associated full forms where present, their control in respect of potential confusion with host language normal lexis, and their appearances in a defined sense dependent situation.

What is novel with our technique is that, with the exception of application specific (enumerable) dictionaries of specific known and controlled abbreviations, controlled acronyms and acronym exceptions, we use intensional case based (equivalence class) techniques based on linguistic criteria with attestations providing decision making traceability rather than extensional ones (such as enumerated dictionaries) for the recognition of potential acronyms.

Our technique is based on micro-systemic linguistic analysis which has a wide applicability in linguistics as for example machine translation (Cardey 2015) and which is underpinned with a formal model (Cardey 2013). Micro-systemic linguistics is the basis for the algorithmic approach leading to the principal document processing algorithm (which involves discerning a partition leading to a classification of acronyms in equivalence classes), but also in the sub-algorithms. For example, in respect of the legal sequences of graphemes in English we observe that we have partitioned the English lexis over 2 and 3 letter sequences contained in each lexis, and in doing so, each non empty (not containing " \emptyset ") cell couple (letter sequence, attestation) gives rise to a distinct equivalence class of English words sharing this letter sequence; the couple being in effect the name of the equivalence class. In particular, hapax attestations correspond to equivalence classes with unit cardinality. Furthermore, it is to be observed that the order of processing for 2 letter sequences and 3 letter sequences is functionally irrelevant; cells corresponding to 3 letter sequences are populated by the linguist establishing and maintaining the table even if the initial 2 letters cell ("plus \emptyset " column) has an attestation (as is the case in the above example with "CHAIN"). No algorithmic optimisation is exercised by the linguist who views each sequence of letters of whichever length in isolation, and this is the correct manner in general to perform such analyses and particularly so here given the tabular ergonomics and thus the declarative table-driven programming involved. This does not preclude subsequent machine executable optimisation over what are in reality nested contexts.

As we have observed, the particular algorithm and the sub-algorithms that we have developed are applicable to a particular industrial context with particular technical documentation working procedures, standards and resources.

On the other hand, we have where possible provided linguistic programming facilities. However, there is little doubt that generalisation to other applications and to other textual material types and contexts would necessitate further research of a linguistic and formal nature and this is what we now show.

4. Part II: Recognising acronyms in general language

In this second principal part of the paper we present a formal model for dealing with general, common English acronym recognition.

A review of the related work on automatic acronym processing has shown a lack of a uniform description of the phenomenon (Schmidt 2016). Proper modelling of acronym-definition systems and the relationship between the pairs would prove useful for devising finely-tuneable algorithms for both the recognition and the *ad-hoc* generation of such pairs. A formal model needs to describe the acronym and its definition on several different levels, at the scope of words, characters, syllables and morphemes.

The purpose of such a formal model is to allow the expression of strict control rules on character and subsequence selection, pairing and substitution. The model also has to encompass the most degenerate cases and still be sufficiently constraining to allow the systematic selection of the correct pairing relation or definition sequence.

4.1. Definitions

An acronym-definition pair is composed of two sequences of characters measurable in either word-length or character-length. Formally, if <A> denotes the acronym-sequence and <D> the definition-sequence, |A| and |D| are their respective length in number of words, and ||A|| and ||D|| their length in number of characters.

4.2. Formal model

Fundamental restrictions on the pair:

- (1) |A| = 1
- (2) |D| ≥ 2
- (2) ||A|| < ||D|| (i.e., an acronym is necessarily shorter than its definition)

Much of the related work constrains the size of candidates relatively to the size of the acronym; such bounds can be formally represented, as for example:

- (3) |D| ≤ j ||A|| + k (j, k positive integers), j then represents the average number of characters represented by a character from the acronym and k represents the maximum number of omitted words (i.e., unrepresented words). We observe that Park and Byrd (2001), Schwartz and Hearst (2003), Nadeau and Turney (2005), and Dannélis (2006) used |D| ≤ min(||A|| + 5, ||A|| x 2) where min(x,y) is the function that selects the minimum value between x and y.

or

- (3') ||A|| ≤ (||D|| / j) (j positive integer).

Taking j = ||D|| / |D| (averaging the distribution of characters) yields:

- (3'') ||A|| ≤ (||D|| x |D|) / ||D||

The description of the pairing relations between subsequences of the elements of the pairs, and more precisely relations between characters requires a finer level of formalism. The first step in building this description consists in indexing the component-characters of each subsequence as follows:

$c_{x,m}$ represents the *m*-th character of the *x*-th word w_x of <D>

c_n represents the *n*-th character of <A>

$c_{x,m} \rightarrow c_n$ signifies that c_n stands for $c_{x,m}$

It is possible that some c_n stands for a longer subsequence in which case we can denote this pairing relation by:

$$\langle c_{x,m}, c_{x,m+1}, \dots, c_{x,m+k} \rangle \rightarrow c_n$$

This representation allows expressing sets of rules representing legal pairing operations within a given acronym system. We take <D> = < w_1, w_2, \dots, w_m > and can then build a matrix representation of the pairing relations for a given acronym-definition pair in the following way:

$$[A_{CRO}]_{ij} = \begin{matrix} n & \text{if } c_{i,j} \rightarrow c_n \text{ (more generally if } c_{i,j} \text{ is in } \langle S \rangle \text{ and } \langle S \rangle \rightarrow c_n) \\ 0 & \text{otherwise} \end{matrix}$$

This matrix being generally very sparse, we use a method similar to the Compressed Space Row (CSR) method of Bank and Douglas (2001) to generate vectors allowing for a more compact description. O ('order') is an *n*-vector containing the non-zero entries of the matrix in row-major order, D ('depth') is a *n*-vector which contain the column-indices of the elements of O and C is a *m*-vector such that C_i is equal to the number of non-zero entries of the *i*-th row. These vectors allow as well the expression of rules and constraints, for example:

- (4) for all O_i : $O_1 = 1, 1 < i - 1 < i \leftrightarrow O_{i-1} < O_i$

Ex.1:

“Acquired Immunodeficiency Syndrome” → “AIDS”

1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
2	0	0	0	0	0	3	0	0	0	0	0	0	0	0	0
4	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

$$r_1 = \langle O = (1, 2, 3, 4), P = (1, 1, 7, 1), C = (1, 2, 1) \rangle$$

To each acronym-definition pair corresponds a unique correct relation among a set of potential ones, and thus has an intrinsic degree of ambiguity. In the case of Ex.1, two other relations can be built respecting the order of the character from the acronym:

$$r_2 = \langle O, P' = (1, 5, 7, 1), C' = (2, 1, 1) \rangle \text{ et } r_3 = \langle O, P'' = (1, 5, 8, 1), C'' = (3, 0, 1) \rangle$$

There is thus a need for stricter constraints on the pairings to choose the correct relation for purposes of representation, recognition or generation. A set of such constraints has to enable us to attribute to each relation a degree of plausibility within a given acronym system. This set can be build *a posteriori* to reproduce and correspond to an observed system or *a priori* in the case of controlled languages for example. The rules are *ordered* and *weighted* so that an algorithm attributing a plausibility score or producing the possible pairings generally will:

- i. match every character from the acronym, in the order that they appear;
- ii. match the initial character from the acronym with the initial character from the initial word of the definition if possible;
- iii. form pairs of characters that conform to at least one rule from a given ruleset **R**;
- iv. select for each pair the highest order rule to which it conforms;
- v. accept only valid acronym-definition pairs within the system defined by **R**;
- vi. reject the acronym-definition pair if no complete pairing can be achieved.

Usually a higher order rule would signify a preference and thus have a bigger weight than lower order ones but nothing prevents to attribute it a smaller or even negative weight.

Example of a simple ruleset **R**¹:

$$\begin{aligned} R^1_1 : c_{1,1} &\rightarrow c_1 & pR(1) &= 4 \\ R^1_2 : c_{x,1} &\rightarrow c_n & pR(2) &= 2 \\ R^1_3 : c_{x,m} &\rightarrow c_n, 1 < m < ||w_x|| & pR(3) &= 1 \end{aligned}$$

where p_x is the weight of the x -th rule and the size of **R**¹ is equal to the number of rules.

These rules function as Boolean-valued functions of the form $R_k((x,m),n)$.

R¹ expressed as functions:

$$\begin{aligned} R1((x,m),n) &= 1 \text{ if } (x=1 \text{ AND } m=1 \text{ AND } n=1 \text{ AND } c(x,m) \rightarrow c(n)), 0 \text{ otherwise} \\ R2((x,m),n) &= 1 \text{ if } (m = 1 \text{ AND } c(x,m) \rightarrow c(n)), 0 \text{ otherwise} \\ R3((x,m),n) &= 1 \text{ if } (1 < m < ||w_x|| \text{ AND } c(x,m) \rightarrow c(n)), 0 \text{ otherwise} \end{aligned}$$

R¹ is deemed 'simple' here mostly because **R**¹₃ has a very broad spectrum. Its purpose is to allow the matching of non-terminal, non-specific characters. We could forego this rule by constructing rules that emphasise the selection of characters located at *boundaries* (i.e. initial word-, syllable-, phoneme- or morpheme- character) (Zahariev 2004). These characters seem to be privileged.

We also want the algorithm to be easily adaptable to output the trace for the solution found or the complete set of solutions. To achieve this goal, we constructed an algorithm capable of reverting to a former successful state if no satisfying solution can be found from the present state and select the next likely state as the new present state. We allow the algorithm to backtrack and retrieve values from a former state as a means to explore alternative paths while always choosing the most desirable next candidate while successively suppressing failed nodes.

4.3. General outline of the recognition algorithm:

- Step 1.** Initialisation (select $c(1,1)$ and $c(1)$, save the initial state).
- Step 2.** Generate all possible matches according to **R** and assign a weight to each one.
- Step 3.** Select the highest-weight, leftmost candidate in $\langle D \rangle$.
- Step 4.** Save the current state, which contains the precedent state.
- Step 5.** Select the next character from $\langle A \rangle$.
- Step 6.** If the end of $\langle A \rangle$ is reached, the algorithm terminates.

Whenever Step 2 yields no possible match, the algorithm reverts to the character selection from the former state and tries the next most-likely candidate. This is achieved by suppressing the failed node.

4.4. Algorithm for recognition

Input:

The indexed sequence $\langle A \rangle$, the indexed sequence $\langle D \rangle$, an ordered set of weighted rules $RP = R \times K \times P_k$

Output:

True if there is a legal relation between $\langle A \rangle$ and $\langle D \rangle$ in **R**, false otherwise

Variables:

(x, m) in $XM = \{1\} \times \{1, \dots, ||w(1)||\} \cup \dots \cup \{ |D| \} \times \{1, \dots, ||w(|D|)||\}$, the index-set for the characters of $\langle D \rangle$

n in $N = \{1, \dots, ||A||\}$, the index-set for the characters of $\langle A \rangle$

k in $K = \{1, \dots, k \mid k = \max(k) : \{k : \text{exists}R(k) = 1\}\}$, the index-set for the rules

$P(n)$ in $P = (XM \times P_k)^N$, a set with a relation $>$ such that :
 $((i,j),p) > ((i',j'),p')$ iff $(p > p' \text{ OR } (p = p' \text{ AND } (i < i' \text{ OR } (i=i' \text{ AND } j < j'))))$, for all $((i,j),p), ((i',j'),p')$ in P
 t in $T = \{0, 1, 2 \dots\}$
 $S(t)$ in $ST = XM \times N \times P \times ST$, space of all possible states

Functions:

existsR: $K \rightarrow \{0,1\}$
 existsR(k) = 1 if R_k exists, 0 otherwise
 pcoord: $P \rightarrow XM$
 $pcoord(p) = pcoord((x,m), pR(k)) = (x,m)$, p in P
 pset: $ST \rightarrow P$
 $pset(S(t)) = pset((x,m), n, P(n), S(t-1)) = P(n)$
 retrieve(S(t)):
 $(x,m) \leftarrow scoord(S(t))$
 $n \leftarrow sn(S(t))$
 $P(n) \leftarrow pset(S(t))$
 $scoord ST \rightarrow XM$
 $scoord(S(t)) = scoord((x,m), n, P(n), S(t-1)) = (x,m)$
 $sn: ST \rightarrow N$
 $sn(S(t)) = sn((x,m), n, P(n), S(t-1)) = n$
 $U(X,Y)$ is the usual set union operation $X \cup Y$

<pre> i. t ← 0 ii. n ← 1 iii. (x,m) ← (1,1) iv. P(n) ← {}. v. S(t) ← ((x,m), n, P(n), ∅) vi. k ← 1 vii. if R(k,(x,m),n) viii.i. P(n) ← U(P(n), { ((x,m), pR(k)) }) viii.ii. if m < w(x) viii.ii.i. (x,m) ← (x,m+1) viii.ii.ii. goto vii. viii.iv. if x < D viii.iv.i. (x,m) ← (x+1,1) viii.iv.ii. goto vii. viii.v. goto xii. ix. k ← k+1 x. if existsR(k) == 1 x.i. goto vii. xi. goto vii.ii. xii. if P(n) == {} xii.i. if n == 1 xii.i.i. return FALSE xii.ii. retrieve(S(t-1)) xii.iii. P(n) ← P(n) - {max(P(n))} xii.iv. goto xii. xiii. (x,m) ← pcoord(max(P(n))) xiv. t ← t+1 xv S(t) ← ((x,m), n, P(n), S(t-1)) xvi. if n < A xvi.i. n ← n+1 xvi.ii. goto vii. xvii. return TRUE </pre>	<pre> # Initialisation t = 0 # Alignment at the beginning of <A> # Alignment at the beginning of <D> # Initializes P(n) as an empty set # Initial state # Start of new character evaluation loop # Evaluate R(k) for the current (x,m) and n positions # Add valid result to P(n) # Check if the end of the current word is reached # Loop back for evaluation of c(x,m+1) ## # Check if the last word of <D> is reached # Loop back for evaluation of c(x+1,1) ## # End of <D> is reached # Increment k # Check existence of R_k # # If no rule is found, takes next character # Empty set means no solution for current state # If there is no match for c(1) # Halting condition # Revert to S(t-1) # Exclude the failed node from P(n) # # Alignment on the highest-order character # Increment t # New state # Check if the end of <A> has been reached # Align to the next character in <A> # # Could also return S(t), the final state, carrying the # trace of the solution </pre>
--	--

The representation of the output of the algorithm applied to Ex.1 is as follows:

