# An Expert System for the Composition of Formal Spanish Poetry

Pablo Gervás
Universidad Europea - CEES, Madrid
pg2@dinar.esi.uem.es
www.uem.esi.es/~pg2

**Abstract:** The present paper presents an application that composes formal poetry in Spanish in a semiautomatic interactive fashion. ASPERA is a forward reasoning rule-based system that obtains from the user basic style parameters and an intended message; applies a knowledge-based preprocessor to select the most appropriate metric structure for the user's wishes; and, by intelligent adaptation of selected examples from a corpus of verses, carries out a prose-to-poetry translation of the given message. In the composition process, ASPERA combines natural language generation and CBR techniques to apply a set of construction heuristics obtained from formal literature on Spanish poetry. If the user validates the poem draft presented by the system, the resulting verses are analysed and incorporated into the system data files.

## 1. Introduction

The automatic generation of text is a well established and promising problem in AI, with numerous practical applications waiting in the sidelines for efficient and acceptable solutions. Existing systems have shown reasonable results in restricted domains [3,7,9], opening the way to considering how more elaborate texts - from the point of view of aesthetics and reader satisfaction - can be obtained [2,4,8]. The composition of poetry ranks among the most challenging problems of language generation, and is therefore a good testbed for techniques designed to improve the quality of generated texts. ASPERA (Automatic Spanish Poetry Expert and Rewriting Application) is a prose-to-poetry semiautomatic translator. By ingenious use of well accepted AI techniques (Natural Language Processing, Case Based Reasoning, Knowledge Based Systems), the application obtains from the user a prose description of the intended message and a rough specification of the type of poem required (length, mood, topic); selects appropriate metre and stanza (by resorting to a knowledge base on literary style); generates a draft of the poem (by applying CBR techniques to a database of previous poems); requests modification or validation of the draft by the user; and updates its own database of information (using NLP techniques to extract all the required linguistic information from the

validated poem). ASPERA is designed to be used as a teaching aid for specific information technology subjects of the degree on Translation and Interpretation at the Universidad Europea - CEES.

# 2. Problem Description

On first acquaintance, the generation of poetry involves advanced linguistics skills and common sense, two of the major challenges that face AI in general. On the positive side, poetry has the advantage of not requiring exaggerate precision. To a certain extent, imposing restrictions over the form of a poem implies a slight relaxation on the specification of the content. Under this assumption, the general problem becomes tractable. There are three main challenges to be faced:

1. a specification of the formal requirements that define a correct poem under classical literary rules (in a format that can be used to drive the construction process)
2. appropriate management of an extensive vocabulary (the choice of words plays an important role in determining the quality of a poem)
3. correct combination of words in the poem to match both the intended message and the chosen metric structure

## 2.1 The Formal Rules of Spanish Poetry

The fact that the application was to be developed in Spanish presents important advantages over other languages. The phonetics of Spanish are quite straightforward to obtain from the written word. Most letters in Spanish sound the same wherever they appear in a piece of text, so the metrics, or the syllabic division, of a verse can be worked out algorithmically [11]. Spanish scholars have a love for rules, and there is a good set of formal rules [10] describing the conditions that a poem must fulfil in order to be acceptable.

The challenge becomes a simple problem of transforming the given evaluation rules (designed to be applied to an existing poem in order to ascertain its acceptability) into the corresponding construction rules.

Given that words are divided into syllables and each word has a unique syllable that carries the prosodic stress, the constraints that the rules have to account for are the following:

1. Specific strophic forms require different number of syllables to a line
2. Syllables may be run together under specific conditions - this is a form of poetic license called *synaloepha* - thereby shortening the syllable count of the line involved
3. The position of the stressed syllable of the last word of a line affects the metric count for that line (adding or subtracting one syllable to the count)
4. Not all possible stress patterns for a line are valid (depending on the length)

5. The rhyme of a word is made up of the last half of its stressed syllable and all following syllables to the end of the word, and each strophic form requires a different rhyming pattern

A poem may be an unstructured sequence of lines, but for the purpose of this application only poems of regular strophic forms are considered. For the purposes of the present application, the following strophic forms are relevant:

1. *romances*, a stanza of several lines of eight syllables where all even numbered lines rhyme together
2. *cuartetos*, a stanza of four lines of eleven syllables where the two outer lines rhyme together and the two inner lines rhyme together
3. *tercetos encadenados*, a longer poem made up of stanzas of three lines of eleven syllables linked together by their rhyme in a simple chain pattern ABA BCB CDC...

These three types have been chosen because each shows a different structural characteristic.

Type 1 presents a recurring rhyme that flows all along the poem. It uses only one rhyme, so many words that rhyme together are required for an acceptable result (our starting data have proven to be poor choices in this respect).

Type 2 presents a very simple but rigid structure. It stands for the simplest possible stanza with enough complexity to be distinguishable from prose (the simplification employed with respect to syntax/semantics makes it difficult for shorter poems to sound acceptable).

Type 3 presents a simple structure that recurs throughout the poem, but with the rhyme changing slowly as it moves down.

A constructive implementation of these rules was developed for the WASP system [4]. WASP was a forward reasoning rule-based system that used a set of construction heuristics obtained from these constraints to produce a poem from a set of words and a set of line patterns provided by the user. The system followed a generate and test method by randomly producing word sequences that met the formal requirements. Output results were impeccable from the point of view of formal metrics, but they were clumsy from a linguistic point of view and made little sense. An improved version of the construction strategies developed in WASP is the starting point of the generating module of ASPERA.

## 2.2 Guiding Word Choice

The set of words available as building blocks for the composition process play a crucial role in determining the quality of the results. Too narrow a choice of vocabulary can lead to formal requirements not being met. Too wide a choice can seriously reduce the efficiency of the system, leading to lengthy searches over vast amounts of words. A trade off must be found: enough words to compose the poem must be available, but they must all be words with a reasonable chance of finding

their way into the final result. This is a part of the problem where intelligent solutions have a good chance of outperforming purely computational techniques.

The ASPID system [5] provided specific algorithms for the selection of a working set of words from an initial vocabulary using methods based on similarity calculations between the message proposed by the user for his poem and a corpus of already validated verses. Based on the similarity calculations, the system established a set of priorities over the complete available vocabulary. The next word to be added to the poem draft was initially looked for only among words marked with the highest priority, with the search extending in subsequent steps to words of lower priority only if none had been found in the previous step. This procedure improved search times considerably and it made possible computations with wider vocabulary coverage and narrower constraints on strophic forms. However, above a certain threshold (of vocabulary size and/or number of constraints imposed on the poem) even the method of establishing a priority ordering on the available words failed to ensure successful termination. The ASPID method of priority assignment is retained in ASPERA, but a prior step of knowledge-based pre-selection of the vocabulary based on user requirements has been added.

## 2.3 Fitting Words to Message and Metric Structure

There are several restrictions on the actual process of composition that must be observed.

The length of a poem is given by the number of lines (and the length of the lines) in the chosen strophic form. The intended message must be shortened or extended to adjust it to the length of the poem.

The basic unit for poem composition is not the sentence but the line. A poem may contain one or several sentences, but it is in its subdivision into lines that the constraints (position of stressed syllables, and rhyme) are imposed. A step of planning is required to distribute the contents of the intended message over the required number of lines.

Words at the end of lines may have additional constraints imposed on them (rhyme) by the chosen strophic form. These restrictions must be taken into account when planning the poem.

The words in a poem must be combined according to the syntax of the language in question, and must make sense according to their semantics. There are two alternative ways of achieving this: to provide the system with adequately rich lexicon, syntax and semantics for the language involved (as done in [8]) for English poetry), or to develop engineering solutions that achieve equivalent results without attempting to model the imposing complexity of the human language faculty. The former solution requires powerful tools capable of representing and manipulating the syntax and semantics of language. For the present application, the latter approach is preferred.

ASPERA resorts to a radical simplification of the linguistic skills underlying poem composition. The exhaustive knowledge approach is abandoned in favour of a heuristic engineering solution. Only the barest outline of a grammatical outline is provided (in the form of a line pattern) to ensure syntactic correctness. Semantic correctness is not enforced strictly, rather an approximate result is obtained by intelligent pre-selection of the choice of words together with a memory based approach to word combination.

The system is provided with a corpus of already validated verses. A Case Based Reasoning approach [1] is applied to this corpus in order to generate new verses. The words in these verses are marked with their corresponding part-of-speech (POS) tag. A line pattern is generated from these POS tags. Each line pattern is a set of tags, and each tag acts as place keeper for a possible word of the line. The tag is actually a string that represents information about part of speech, number, and gender of the word that would stand in that particular place in the pattern. Patterns act as seed for lines, therefore a pattern determines the number of words in a line, the particular fragment of sentence that makes up the line, and the set of words that can be considered as candidates for the line. By following this heuristic shortcut, the system is able to generate verses with no knowledge about grammar or meaning.

# 3. ASPERA: Automatic Spanish Poetry Expert - a Rewriting Application

ASPERA is a forward reasoning rule-based system that performs the following sequence of operations:

1. interacts with the user to obtain a specification of the desired poem (intended message, mood, setting);
2. searches its knowledge base to find the most appropriate strophic form to match that specification;
3. plans a poem draft by distributing the intended message over the chosen strophic form;
4. pre-selects and loads a task-specific vocabulary and corpus of verse examples for each fragment of the poem by adequately combining its data files (CBR Retrieve step I);
5. generates each of the lines of the poem draft by mirroring the POS structure of one (CBR Retrieve step II) of the pre-selected line examples combined with the words in the pre-selected vocabulary (CBR Reuse step);
6. presents the draft to be validated or corrected by the user (CBR Revise step); and
7. carries out a linguistic analysis of any validated poems in order to add the corresponding information to its data files to be used in subsequent computations (CBR Retain step).

ASPERA is written in CLIPS, a rule-based system shell developed by NASA. Earlier implementations were attempted using Prolog, but the nature of the line generation part of the problem, being a constructive combination of a set of elementary ingredients towards the achievement of a single whole which is the final poem, lends itself more easily to a forward-reasoning mode of operation. The system's poetry expert knowledge base had originally been coded in Prolog, but translation onto the new paradigm was presented no special problems.

## 3.1 Collection of Poem Specifications

The system interacts with the user to obtain specifications of the desired poem. The user is first asked a set of questions designed to ascertain a few elementary facts about his intentions. These will be used by the knowledge-based module of the system to select the right vocabulary and an adequate set of verse examples from the available corpus.

At present, the system is designed to operate with the following parameters:

- *approximate length of the poem*: The user has to provide a numerical value for the number of lines he would like to obtain. The system uses this number in working out an adequate strophic form (or combination of strophic forms) for the poem. The user is also asked whether the stated length is a rigid or flexible constraint.
- *rhyme structure*: Some strophic forms have rigid rhyme structures (there are constraints on the rhyme of every line) and others are more flexible (there are constraints on only some of the lines).
- *degree of formality*: Certain strophic forms are more formal than others. This distinction plays an important role in selecting the appropriate form.
- *setting*: The system provides a choice between an urban setting or a rural setting. This is a very restricted choice, but it may be extended at later stages.
- *mood*: As a first approximation, mood is interpreted either as positive or negative.

Length of poem and degree of formality are used specially to determine the strophic form. Setting and mood are used to select the correct vocabulary.

Once the basic parameters have been set, the user is asked to provide a prose paraphrase of this intended message. This paraphrase need not be grammatically sound, and may range from a full explicit text to a set of keywords that the user would like to show up in the poem. The intended message is stored as a list of words in a format amenable for later use by the system:

```
(message Peter loves Mary they go together to the beach)
```

Every word in the message is considered a good candidate to contribute to the final poem. Statistical methods of natural language processing applied in the field of information retrieval favour an initial trimming of user queries with a view to retaining for processing only terms that have relevance for the retrieval task. Stop

lists are applied in order to eliminate empty words (pronouns, articles, prepositions...). For the present purposes, however, the presence of words of these types allows the user to control the style and appearance of the final result even more (or at least as much as) nouns, verbs and adverbs.

## 3.2 The Poetry Expert Knowledge Base

The Poetry Expert knowledge base contains two distinct sets of rules that encode specific knowledge concerning the relationship between strophic forms, the user's wishes, and the available data file. Their job is to select the best available combination of strophic form and data files to suit the user's request.

The first set of rules concerns the choice of strophic form and the selection of data files with the required sets of verse examples/patterns. They have been obtained by a knowledge acquisition process performed systematically over the author's personal experience, careful analysis of classical Spanish poetry, and available references on poetic analysis [10]. This gave rise to a complex set of rules that relate possible combinations of input parameters with possible outputs (strophic forms or combinations of strophic forms). The relationship is mostly associative, but requires a step of arithmetic calculation when processing the required length. The decisions embodied in the rules follow roughly the following guidelines:

- formal poem → strophic forms of 11 syllables.
- informal → strophic forms of 8 syllables.
- long and flexible → *romance* or *terceto encadenado*
- short and concise → *terceto*
- fully rhymed → *cuarteto* or *terceto encadenado*
- loosely rhymed → *romance* (long) or *terceto* (short)
- many similar rhymes → dos *cuartetos* (not too long) or *romance* (long)

The second set of rules associates the information obtained from the user regarding setting and mood desired for the poem with the various data files over which the available vocabulary is distributed.

The application of the Poetry Expert knowledge base results in a message to the user proposing a specific strophic form. If the user accepts the proposal, the corresponding data files are loaded and the system moves onto the next stage. If the user rejects the proposal, the system allows the users choice of strophic form to override its own suggestion. The knowledge base is then addressed for the right combination of data files and these are loaded. Mismatching specifications of poem length or degree of formality and choice of strophic form may lead to non termination.

### 3.3 The Elementary Planning Step

Given the user's choice of basic parameters the system uploads an appropriate combination of data files containing vocabulary extracts. A vocabulary entry for a word is a CLIPS fact of the form:

```
(word (cual desde_n) (numsil 2) (accent 2)
      (emp 0) (term 0) (cat NCMS)
      (rima en) (level nil))
```

An entry must provide all the necessary information for imposing the metric restrictions: length of the word in syllables (numsil), position of stresses syllable (accent), whether the word starts (emp) or ends (term) with a vowel, the POS tag for the word (cat), and the rhyme (rima). An additional level field is provided to enable priority marking of the words during later processing.

For the resulting choice of strophic form specific line patterns /examples are loaded. A line pattern/example is a CLIPS fact of the form:

```
((sample (n 72)
         (patt NCMS PDEL NCMS NCMS DET PPO3FS NCFS)
         (words desde_n del cielo error de la ventura)
         (beg 0) (end 1) (level nil))
```

where n is a unique identifier for the specific sample, patt is the set of POS tag corresponding to the sample - which acts as line pattern -, words is the actual line example, beg encodes information about whether such line starts a sentence, end encodes information about whether the line ends a sentence, and the level field is used for priority settings.

The words in the intended message provided by the user are distributed into as many fragments as there are lines in the chosen strophic form, retaining within each fragment the original order of appearance. For instance, supposing the message presented above were to be rewritten as a terceto, it would be split by the system into the following facts:

```
(fragment 1 Peter loves Mary)
(fragment 2  they go together)
(fragment 3 to the beach)
```

This is not considered a draft of the poem, but only as an approximate distribution of information over lines of the poem.

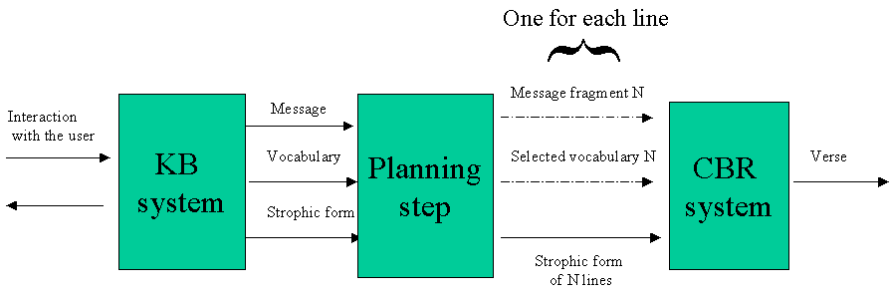


**Figure 1 Basic structure of the ASPERA system**

A module of the system works out the similarities between the line fragments and the available line examples/patterns. Each fragment/pattern pair is indexed with a numerical value indicating their similarity. In the initial approximation, this similarity value is worked out as the number of POS tags they have in common. The results of these similarity calculations are later used during the CBR Retrieve step.

Another module counts the number of rhymes available (if any) for each of the pre-selected words. These numbers are later used to assign specific words and patterns to each fragment during the CBR Retrieve step: a pattern is only good for a given line in the poem if there is a word of the necessary rhyme whose POS matches the last POS tag in the pattern.

The CBR Retrieve step can be considered to start during the planning stage, since line patterns/examples are assigned to the different fragments of the poem at this stage. This is because the retrieval of resources for one line is not independent from the assignation of resources to its neighbours. It is important, for instance, to ensure that consecutive lines are assigned sets of patterns that allow a correct linguistic fit between the resulting lines (the sentence fragment in the first line can be continued with the sentence in the following one, or there are patterns for starting a sentence on the following line if patterns in the first line include an end of sentence at the end of the line).

## 3.4 The CBR Approach to Line Generation

A different CBR process is set in motion for each line in the poem. This is done sequentially starting from the first line, but none of the different CBR processes progresses onto its next stage until all the other processes are ready to do so as well: all processes retrieve their vocabularies and line examples, all processes adapt their selection to generate a new line (the revision and analysis phases can be done in any order). This ensures that no two lines in the same poem are built following the same pattern and that words are not repeated in a poem.

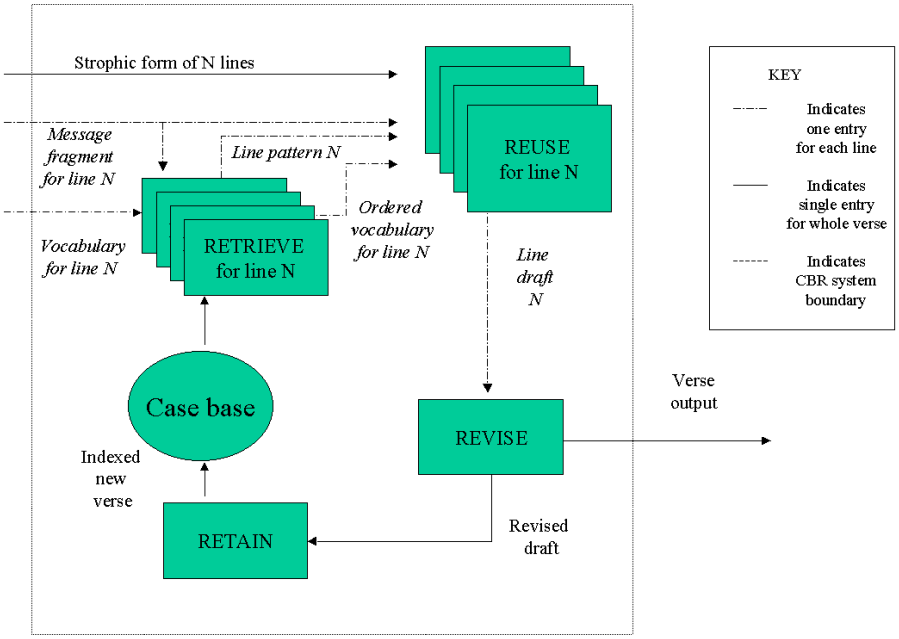


**Figure 2 Verse generation using CBR**

### 3.4.1 Retrieve step

For each fragment of the poem, adequate words and adequate line patterns/examples must be selected. Line examples/patterns have already been assigned during the planning stage.

A module assigns priorities to all the available words, so that only the most adequate words are considered for each line. The criteria used in this assignation are, in decreasing order of priority: words actually appearing in the corresponding fragment of the intended message, words with the required rhyme for the line.

These priorities can be modified once the initial lines have been generated, to include any words in the fragment of the intended message corresponding to previous lines that have not been used in the generation of the actual verse.

### 3.4.2 Reuse step

For each set of fragment of the intended message, set of line examples /patterns, and set of selected words, the basic generation algorithm following the constructive rules for the appropriate line length is applied.

The elementary generation units are the *line pattern* being followed (which acts as guiding form) and the *draft of the current line*. At each step of the generation the words are chosen to match the next POS tag in the line pattern (always according to their established priority ordering). If the chosen word meets the constructive requirements, it is appended to the draft of the current line and the process is

iterated. The constructive requirements are implemented in the form of imperative Boolean functions.

If no valid extension is found, the system allows limited backtracking: the last word to be added can be removed and an alternative attempted. An additional process of annotation takes place to avoid a repetition of alternatives that have already been tried.

Additionally, the assignation of line examples/patterns to each fragment of the poem can be revised during the Reuse step if no solution is found with the existing assignation once all possible backtracking alternatives over the assigned words have been exhausted. If the assignation of line example/pattern is modified, the assignation of words may have to be revised. This new word assignation is now not only constrained by the words assigned to previous fragments, but also by the words already assigned to following fragments. However, if the generation process for any previous line has already finished at that point, any words of its fragment of the intended message that have not been used can be considered.

### 3.4.3 Revise step

As soon as all the generation processes have concluded, the whole poem draft is presented to the user. The user is required to validate each of the lines. If the user comes up with possible modifications, the system accepts them instead of the generated verse. The modified versions suggested by the user ought to be tested for metric correctness. This feature is not currently contemplated, under the assumption that any corrections the user makes will either be well founded or they will be in repair of a serious syntactic or semantic error introduced by the heuristic approximation.

### 3.4.4 Retain step

All verses validated by the user are processed to create a personal data file containing the corresponding line examples/patterns in the correct format. Any new words introduced by the user either as part of the intended message or modifications of the proposed draft are also added to the data. This will ensure that regular use of the system will improve the quality of results only if the degree of satisfaction with the initial poems is reasonable high. This implies that although the system is capable of a certain amount of learning once a reasonable standard (of vocabulary and stored examples) has been achieved, an initial threshold of vocabulary and corpus adequacy must be met for the system to function adequately.

## 3.5 ASPERA Results: Examples

The following examples illustrate system input and resulting poem for specific instances of use of the ASPERA system. Suppose the user expressed a desire for a short, fully rhymed, formal poem, with a rural setting and a positive mood. The following specification was given to guide the search in example 1:

*tan hermoso viento fue corazón mudo*

The system suggested a *terceto* should be attempted (lines 11 syllables long rhyming ABA), and provided a set of model poems from its case base. Model poems were of the form given in example 0:

*Sabed que en mi perfecta edad y armado*
*con mis ojos abiertos me he rendido*
*al niño que sabéis ciego y desnudo.*

**Example 0. Model poem**

Since rhyme restrictions are imposed by the generation mechanisms independently of the model poems, these need not follow the rhyming pattern required. The model poems provide guidance as to the number and specific POS elements to be used in constructing each line. Vocabulary files appropriate to the setting and mood where loaded and the system produced the following poem:

*Ladrará la verdad el viento airado*
*en tal corazón por una planta dulce*
*al arbusto que volais mudo o helado.*

**Example 1.**

Three -*viento, corazón* and *mudo* - of the six words in the specification have made it into the twenty words of the final verse. Six words of the poem originate in the selected vocabulary -*ladrará*, *planta*, and *arbusto* through the rural requirement and *dulce*, *verdad*, and *volais* from the positive mood. Two words - *airado* and *helado* - come from the model poems. These words are forced to appear in the poem by the additional restrictions posed by the rhyme on the final words of the first and last line. As inexperienced human poets do, ASPERA simply chooses two words that rhyme and happens to find more of those among the model poems[1]. The rest of the words form part of ASPERA's elementary vocabulary and are used to hold the poem together with an approximation of sense. In this particular case, the last line of the resulting poem closely follows the structure of the last line of the sample model poem presented. The structure of the other lines is based on different model poems, chosen by the planning stage of the system to ensure minimal overall coherence of the poem.

A similar analysis can be carried out for example 2:

*Andando con arbusto fui pesado*
*vuestras hermosas nubes por mirarme*
*quien antes en la liebre fue templado.*

**Example 2.**

resulting from similar general choices and the specification:

---

[1] This behaviour may improve if enough rhyming words are provided either in the selected vocabulary or the specification.

*andando por el monte hermosas nubes una liebre*

The poem as it stands is quite obscure, however, it can be seen very clearly in this case how the specification and the motives - rural and positive - drive the system.

# 4. Benefits to be Expected from ASPERA

ASPERA may prove beneficial to a certain extent in two different contexts. Within the restricted domain of tuition in the field of literature, and more widely where it may shed light on general language problems.

### 4.1.1 As a Pedagogical Tool

In the long run, it is hoped that ASPERA will provide guidelines to develop better pedagogical tools for the field of Spanish poetry. For the average person, a poem is either pleasing or not, and the introduction of - sometimes very complex - rules to be computed as part of its enjoyment goes against the grain on first time acquaintance. ASPERA provides the means for students to see how the imposition of the formal rules may affect the form of a message of their choosing, without requiring them to have learnt all possible combinations of the rules. It is expected that interacting with the system may help the students to develop - without learning the formal rules - the instinctive feel for 'correctness' of a verse that a poet achieves naturally.

### 4.1.2 As a Potential Source for Slim Language Processing Heuristics

The solutions applied in ASPERA to language problems are all highly heuristic in nature, and resort to no complex linguistics techniques. The examples presented show the advantages and the shortcomings of the approach: message content gets scrambled for the sake of form, yet applications relying very heaviliy on specific formats and not requiring originality may find the simplicity of the techniques compensates for the loss of precision. From this point of view the ASPERA system, as it is evolved to more refined heuristics solutions to language generation, constitutes a benchmark for simplified approaches.

# 5. Conclusion

ASPERA is the computing heart of an application with a great potential for user satisfaction. In its current version, it is handicapped by clumsy interfaces that are unable to compete with state of the art GUI technology. However, the underlying methodologies and techniques have shown their adequacy to the task even in their budding state of development. The heuristics techniques applied to resolve linguistic problems at all levels - syntactic and semantic - open interesting avenues of research in language processing. The system is open to enhancement and adaptation once user feedback has been gathered during evaluation.

## References


[1] Aamodt, A. & Plaza, E. (1994). Case-Based Reasoning: Foundational Issues, Methodological Variations, and System Approaches. AI Communications, 7(i), pp.39-59.

[2] Bailey, P., 'A reader-based model of story generation', Symposium on Artificial Intelligence and Creative Language: Stories and Humour, AISB'99 Convention, 6th-9th April 1999, Edinburgh College of Art & Division of Informatics, University of Edinburgh.

[3] Chevreau, K, Coch, J., García Moya, J.A., and Alonso, M., 'Generación multilingüe de boletines meteorológicos', Procesamiento del Lenguaje Natural, SEPLN, N. 25, September 1999, pp 51-58.

[4] Gervás, P., 'WASP: Evaluation of Different Strategies for the Automatic Generation of Spanish Verse', in: Time for AI and Society, Proceedings of the AISB-00 Symposium on Creative & Cultural Aspects and Applications of AI & Cognitive Science, 17th-18th April 2000, University of Birmingham, England, pp 93-100.

[5] Gervás, P., 'Un modelo computacional para la generación automática de poesía formal en castellano', in: Actas del XVI Congreso de la SEPLN (Sociedad Española para el Procesamiento del Lenguaje Natural), Vigo, España, September 2000.

[6] Gervás, P., 'A Logic Programming Application for the Analysis of Spanish Verse', in: Lloyd, J., et al, Computational Logic - CL 2000, First International Conference, London, UK, July 2000, Proceedings, Springer Verlag.

[7] Horacek, H. and Busemann, S., 'Towards a Methodology for Developing Application-Oriented Report Generation', in: Günter, A. and Herzog, O. (eds.), 22nd German Conference on Artificial Intelligence (KI-98), Proceedings, Bremen, Germany, 1998.

[8] Hisar Maruli Manurung, Graeme Ritchie and Henry Thompson, 'Towards a computational model of poetry generation', in: Time for AI and Society, Proceedings of the AISB-00 Symposium on Creative & Cultural Aspects and Applications of AI & Cognitive Science, 17th-18th April 2000, University of Birmingham, England.

[9] Nederhof, M.-J., 'Efficient generation of random sentences', Encyclopaedia of Computer Science and Technology, Vol.41, Marcel Dekker, 1999, pp 45-65.

[10] Quilis, A., (1985) 'Métrica española', Ariel, Barcelona.

[11] Real Academia Española (Comisión de Gramática), (1986) 'Esbozo de una nueva gramática de la lengua española', Espasa-Calpe, Madrid.

[12] Sánchez León, F., Corpus Resources And Terminology ExtRaction project (MLAP-93/20), 'Spanish tagset of the CRATER project', ftp://ftp.lllf.uam.es/pub/CRATER/esT-tagger-1-0.tar.gz