An Electronic Assistant for Poetry Writing

Nuno Mamede*, Isabel Trancoso*, Paulo Araújo†, and Céu Viana♮
{*Isabel.Trancoso, Nuno.Mamede, Paulo.Araujo, Ceu.Viana*}*@l2f.inesc-id.pt*

* INESC-ID Lisboa/IST †INESC-ID Lisboa/ISEL ♮ CLUL
L^2F - Spoken Language Systems Lab
INESC ID Lisboa, Rua Alves Redol, 9,1000-029 Lisboa, Portugal
http://www.l2f.inesc.pt

Abstract. The ultimate goal of the poetry assistant currently under development in our lab is an application to be used either as a poetry game or as a teaching tool for both poetry and grammar, including the complex relationships between sound and meaning. Until now we focused on the automatic classification of poems and the suggestion of the ending word for a verse. The classification module is based on poetic concepts that take into account structure and metrics. The prediction module uses several criteria to select the ending word: the structural constraints of the poem, the grammatical category of the words, and the statistical language models obtained from a text corpus.
The first version of the system, rather than being self-contained, is still based on the use of different heterogeneous modules. We are currently working on a second version based on a modular architecture that facilitates the reuse of the linguistic processing modules already developed within the lab.

1 Introduction

This paper describes the early stages of the development of a system to assist poetry writing. Its main goals are the classification of poems and the suggestion of verses' ending words. The poem classification is based on poetic concepts, such as the number of verses in the stanzas; the stanzaic form; the number of syllables on the verses; and the rhyme scheme. The suggestion of the ending word of a verse may be based on different selection criteria or combination of criteria: the structural constraints of the poem, the grammatical category of the words, and the statistical language models obtained from a text corpus.

With such a system, we intend on one hand to help understanding the structure and rhyme of poems, which may be important to improve the reading aloud capabilities of students and, on the other hand, to encourage them to write their own poems.

Although there are some Internet sites that publish and discuss poetry for Portuguese [1, 2], there are no computer programs to assist the process of writing poems for our language and we could not find any framework that handles rhyme correctly. A rhyme dictionary [3] exist, but it only takes into account the final letters of the words, not their pronunciation.

For the English language, we could find some interesting word games where the user can make poems with those words [4], programs that allow the generation of nonsense poems based on syntax definitions [5], and others that generate poetry based on models created from poetry corpora [6].

One of the interesting features of our application is the fact that it relies on tools which, with few exceptions, have already been implemented in the lab. Hence, the first version of the system, rather than being self-contained, is still based on the use of different heterogeneous modules. We are currently working on a second version based on a modular architecture that facilitates the reuse of the already existent linguistic processing modules.

We shall start the description of our poetry assistant system with the classification module (Section 2), focusing on the submodule that presents some important research challenges - the metric syllable counter. Section 3 describes the prediction module, and the selection criteria that can be combined to suggest the missing words. Finally, Section 4 describes the new system architecture we are currently working on.

The two following sections report informal evaluation experiments conducted using a very small test corpus that was manually classified. In addition to around 20 stanzas from two very well known Portuguese poets, the corpus also includes around 200 poems written by children (ages 7-9). The corpus collection task is by no means complete yet. In fact, it has recently been enlarged with the poems collected in the scope of a national project dealing with digital spoken books [7]. The informal tests described below, however, do not yet include this recent addition.

2 Classification module

This module currently takes as input a finished poem, and yields as output the number of lines and stanzas, the stanzaic form, and the rhyme scheme.

One of the main tools used by this module is a Grapheme-to-Phone conversion tool. Several approaches have been developed in the Lab for this purpose: based on rules, neural networks [8], and classification and regression trees [9]. The current GtoP module of the DIXI+ system is based on CARTs trained on a lexicon and has a word error rate of 3.8%, slightly higher than the rules system. Our latest efforts in terms of GtoP conversion have expressed the original rules as FSTs [10].

The GtoP tool yields the phonetic transcription of each verse, taking into account generic word co-articulation rules. It outputs a string of symbols of the SAMPA phonetic alphabet for European Portuguese [11], as well as lexical stress markers.

The following is an example of the output of the classification module, using the first stanza of the most famous epic poem in Portuguese (Os Lusadas, by Camões, XVI century):

As armas e os barões assinalados
Que da ocidental praia lusitana
Por mares nunca dantes navegados
Passaram ainda além da Taprobana,
Em perigos e guerras esforçados
Mais do que prometia a força humana,
E entre gente remota edificaram
Novo Reino, que tanto sublimaram;

```
Summary:
  lines: 8
  verses: 8
  stanzas: 1
  rhyme: [A,B,A,B,A,B,C,C]
  syllables: [10,10,10,10,10,10,10,10]
  Classification: ‘‘Ottava rima’’
```

Although the relevance of the phonetic transcription tool for the rhyme classification module is not well illustrated by this example (where the same conclusions over rhyme could be derived from the orthography alone), this is not always the case. The above illustrated rhyming criterion requires perfect rhymes, where the last stressed vowel and succeeding sounds are identical. An alternative criterion could be rhyming only in terms of the two last syllabic nuclei (e.g. *dentro* and *vento*, *silêncio* and *cinzento*).

2.1 Metric structure

One of the main research challenges of this project is dealing with the metric structure of the poems. In fact, counting metric syllables (up to the last stressed syllable) is not an easy task, if one takes into account word co-articulation phenomena and different possibilities of phonological reduction.

In the current version of the classification module, syllabic boundary markers are placed a posteriori on the SAMPA string using a simple script based on generic rules. As an example of its application to the sixth verse in the above poem, we obtain the correct number of 10 metric syllables (syllable boundaries are marked by ”$” or by `"`, for stressed syllables), accounting for vowel coalescence and diphthonguisation (7th and 9th syllables, respectively).

```
"majZ$du$k@$pru$m@"ti$a"for$s6w"m6$n6
```

We are currently working on another version which allows for some phonological variation not accounted for by the GtoP rules, yielding for each verse a range of values, rather than a single value, for the number of metric syllables. This work is closely related with our past work on pronunciation variation for phone alignment [12] where we have modeled variants that depend on the local immediate segmental context through rules implemented via finite state transducers. The main phonological aspects that the rules are intended to cover are: vowel devoicing, deletion and coalescence, voicing assimilation, and simplification of consonantal clusters, both within words and across word boundaries. Some common contractions are also accounted for, with both partial or full syllable truncation and vowel coalescence. Vowel reduction, including quality change, devoicing and deletion, is specially important for European Portuguese. In fact, as a result of vowel deletion, rather complex consonant clusters can be formed across word boundaries.

Notice that vowel deletion can be deliberately marked in the orthography by the authors themselves (e.g. *já como um ‘stigma* by David Mourão Ferreira in “Dos anos Trinta”, in which the first vowel “e”, pronounced as a schwa and often deleted, is not included in the orthography).

2.2 Meter

The identifi cation of the stressed syllables is also important for classifying the meter - rising (iambs and anapests), or falling (trochees and dactyls), depending on having one or two unstressed syllables followed by a stressed one or a stressed syllable followed by one or two unstressed syllables. It is also important for classifying each verse according to the position of the stressed syllable of the last rhyming word (oxytone words have the accent in the last syllable; paroxytone ones have the accent in the penultimate syllable; and proparoxytone words have the accent in the antepenult syllable). Proparoxytone verses are much less frequent.

The meter classifi cation part is not yet implemented.

2.3 Informal evaluation

An informal evaluation of the classifi cation module was done using our still small test corpus. The results obtained with the already implemented modules are very positive.

A byproduct of this work was the development of the fi rst electronic Portuguese rhyme dictionary, in which the search for rhyming words is made based on the word's phonetic transcription.

3 Word prediction module

This module takes as input an incomplete poem, for which the user may request the suggestion of a word that rhymes with the other verses of the stanza. The suggestion may be based on different selection criteria or combination of criteria, previously defi ned by the user. The selection criteria should include:

– words rhyming with the last verse
– words rhyming with a given verse
– metrical atructure of the verse
– words belonging to a given grammatical category (noun, adjective, verb, ...)
– words belonging to a given ontology (animal, tree, flower, job, country, ...)
– words that may follow the last word written by the user, according to some statistical language model

The list of words that satisfy all the criteria defi ned by the user may be too large to allow its presentation to the user. Hence the module selects the 10 best words in this list.

The prediction module uses the same GtoP and metric syllable counting tools as the classifi cation module. In addition, it uses some linguistic analysis modules already available in the lab, for the generation of possible word classes that may follow the last word of the incomplete poem:

– Smorph: Morphological Analyser [13]
– PAsMo: Post-Morphological Analysis [14]
– SuSAna: Surface Syntactic Analyser [15]

The submodule related to the ontology criterion is not yet implemented. The submodule that involves statistical language models uses n-gram models built using the CMU-Cambridge Statistical Language Modeling toolkit [16] which provides support for n-grams of arbitrary size, and support for several discounting schemes. Rather than retraining new models with our restricted poetry corpus, we used the existent n-grams currently built for a broadcast news task (57k unigrams, 6M bigrams, 11M trigrams, ...).

As an example of the application of the word prediction module, consider the following poem by António Aleixo:

A quem prende a água que corre
é por si próprio enganado;
O ribeirinho não morre,
vai correr por outro lado.

Let us suppose that the last word (*lado*) was not included. By using a bigram model criterion, the first 10 words suggested would be (by decreasing order of frequency):

lado, dos, a , de, que, o, para, e, dia, em, ...

On the other hand, by using a rhyme criterion, the first 10 suggested words would be:

lado, passado, resultado, dado, deputado, avanado,
machado, demasiado, obrigado, advogado, ...

The application of the number of metric syllables criterion (7 in this case, for the other verses), together with the above criteria, restricts the list of 2-syllable suggested words to:

lado, dado, fado

The missing word was thus successfully suggested by the prediction module.

3.1 Informal evaluation

An informal evaluation of this module was conducted using our still very small poetry corpus as test corpus and repeating the above procedure of removing the last word of each stanza. Our preliminary results led us to conclude that n-grams of higher order (3-grams, 4-grams, ...) are not very effective, as the missing word is not always among the first 10 most frequent n-grams, and in some cases it is not present at all.

The effectiveness of the use of structural constraints was also compared with the use of the bigram models. Our first tests intended to compare the bigram criterion with the one using the number of metric syllables. Our preliminary results led us to conclude that in the 10 most frequent words produced by the second criterion, there are more words that could be used as a substitute of the missing word than in the corresponding list produced by the bigram criterion. Similar preliminary conclusions could be drawn when comparing the use of the bigram criterion with the rhyme one: the latter yielded a larger number of words that could be used as a replacement of the missing word. The last bunch of tests used a combination of the two structural constraints: number of syllables and rhyme. In this case, most of the words in the 10-best list could be used as a replacement.

4 System architecture

As stated above, one of the interesting features of our poetry assistant is the reuse of several already available linguistic processing modules. Hence it was particularly important to choose a suitable architecture. We wanted it to be domain independent, language independent, distributed (over the internet) and normalized over each type of functional module. Moreover, we would like to have the possibility of aggregating existing modules to obtain a ”new” module with a ”new” interface.

These design goals led to the development of GalInHa [17], a web-based user interface for building modular applications that enables users to access and compose modules using a web browser. GalInHa is the short name for Galaxy Interface Handler, inspired as the name indicates on GALAXI II [19], the architecture defined at MIT and used by the DARPA Communicator initiative. GALAXI II is a distributed system with a client-server architecture. It concentrates the communication between modules in a single module, the hub, using a standardized protocol.

Since GalInHa deals with Galaxy process chains, all that is needed for the development of the poetry assistant is to define the corresponding chain and to connect the modules. Some of the needed linguistic analysis modules are already available through GalInHa (E.g. Smorph, PAsMo, SuSAna), although they accept/produce different data formats, so additional modules were needed to provide data format conversion.

All that is needed to connect the new modules is to include an existing XSLT [20] processor into GalInHa. For that, we have to write a wrapper to call external applications. The other modules are currently being adapted to the GalInHa architecture.

`Galinha`'s Galaxy-based general architecture is shown on figure 1.

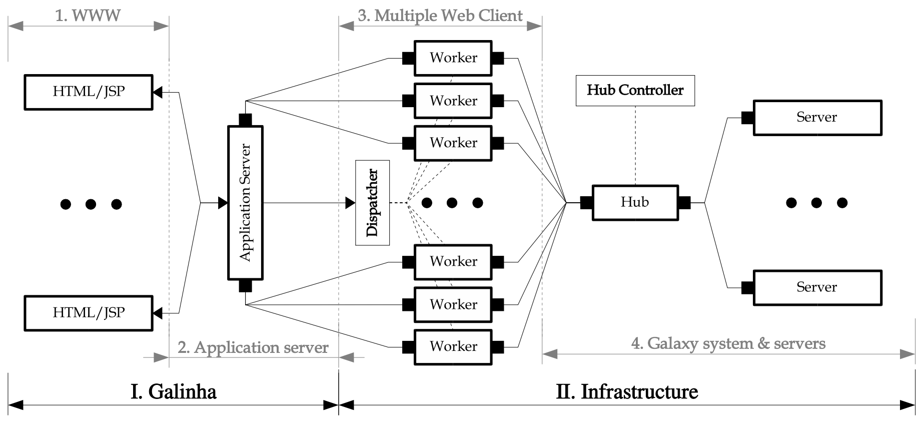

Fig. 1. The Galaxy infrastructure, control servers, and user-side levels.

The application server is one of the interface's key components. It provides the runtime environment for the execution of the various processes within the web application. Moreover, it maintains relevant data for each user, guarantees security levels and manages access control. The interface also uses the application server as a bridge between the interface's presentation layer (HTML [24]/JavaScript [21], at the browser level) and the infrastructure.

The presentation layer consists of a set of PHP scripts and classes [22] and related pages. It is built from information about the location (host and port) of the MultipleWebClient that provides the bridge with the Galaxy system the user wants to contact; and from XML [23] descriptions of the underlying Galaxy system (provided by the hub controller). This is a remake of a previous Java/servlets-based interface.

Besides allowing execution of services on user-specified data, the interface allows users to create, store, and load service chains. Service chains are user-side service- or program sequences provided by the servers connected to the infrastructure: each service is invoked according to the user-specified sequence. Service chains provide a simple way for users to test sequences of module interactions without having to actually freeze those sequences or build an application. The interface allows not only inspection of the end results of a service chain, but also of its intermediate results. Service chains may be stored locally, as XML documents, and may be loaded at any time by the user. Even though, from the infrastructure's point of view, service chains simply do not exist, selected service chains may be frozen into system-side programs and become available for general use. Other developments on the user side are currently being studied to address sharing of chain configurations without infrastructure support.

Modules may be included in the infrastructure in two ways: the first is to create the module anew or to adapt it so that it can be incorporated into the system; the second is to create a capsule for the existing module – this capsule then behaves as a normal Galaxy server would.

Whenever possible or practical, we chose the second path. Favoring the second option proved a wise choice, since almost no changes to existing programs were required. In truth, a few changes, mainly regarding input/output methods, had to be made, but these are much simpler than rebuilding a module from scratch: these changes were caused by the requirement that some of the modules accept/produce XML data in order to simplify the task of writing translations. This is not a negative aspect, since the use of XML as intermediate data representation language also acts as a normalization measure: it actually makes it easier for future users to understand modules' inputs and outputs. It also eases the eventual conversion between module's outputs and inputs (also needed on a command-line approach).

5 Conclusions and future trends

This paper reported on-going work on the development of a poetry assistant system which, besides involving many already existing tools in the lab, also presents some interesting research challenges not only in terms of the overall architecture, but namely in terms of metric structure. The nest stage of the project will focus on rhythm and intonation and on how these may convey differences of meaning.

The last stage of the project will be devoted to evaluation. We plan to enlarge our test corpus in order to conduct some formal evaluation of the two modules (also assessing response time). In addition, we plan to do some user evaluation, using a panel of primary and high-school teachers and students, in order to evaluate the performance and the usability of the interface.

We hope that this system would be a valuable e-learning tool which may be used in classrooms or at home, to classify poems, help understand the concepts of structure and rhyme, and encourage students to write their own poems by suggesting the next words.

References

1. *Geraç ˜ao Poesia*, World Wide Web Consortium (W3C), 2001, see: http://www.geracaopoesia.meublog.com.br.
2. *Projecto Vercial*, 1996-2004, see: http://www.ipn.pt/opsis/litera.
3. *Dicion´ario de Rimas Po´eticas*, Pretor Inform´atica e Sistemas Ltda, 2000, http://www.lemon.com.br.
4. *Magnetic Poetry Kit*, Magnetic Poetry, Inc., 2000, see: http://www.magneticpoetry.com.
5. A. Chachanashvili, *Dada Poem Generator*, 2000.
6. R. Kurzweil, *Ray Kurzweil's Cybernetic Poet*, CyberArt Technologies, 1999, see: http://www.kurzweilcyberart.com/poetry/rkcp-overview.php3.
7. A. Serralheiro, I. Trancoso, D. Caseiro, T. Chambel, L. Carriço, and N. Guimar˜aes, "Towards a repository of digital talking books," in *Proc. Eurospeech '2003*, Geneva, Switzerland, Sept. 2003.
8. I. Trancoso, M. Viana, F. Silva, G. Marques, and L. Oliveira, "Rule-based vs. neural network based approaches to letter-to-phone conversion for portuguese common and proper names," in *Proc. ICSLP '94*, Yokohama, Japan, Sept. 1994.
9. L. Oliveira, M. C. Viana, A. I. Mata, and I. Trancoso, "Progress report of project dixi+: A portuguese text-to-speech synthesizer for alternative and augmentative communication," FCT, Tech. Rep., Jan. 2001.
10. D. Caseiro, I. Trancoso, L. Oliveira, and C. Viana, "Grapheme-to-phone using finite state transducers," in *Proc. 2002 IEEE Workshop on Speech Synthesis*, Santa Monica, CA, USA, Sept. 2002.
11. *SAMPA (SAM Phonetic Alphabet)*, Spoken Language Systems Lab (L2F), see: http://www.l2f.inesc-id.pt/resources/sampa/sampa.html.
12. I. Trancoso, D. Caseiro, C. Viana, F. Silva, and I. Mascarenhas, "Pronunciation modeling using finite state transducers," in *Proc. 15th International Congress of Phonetic Sciences (ICPhS'2003)*, Barcelona, Spain, Aug. 2003.
13. S. Ait-Mokhtar, "L'analyse pr´esyntaxique en une seule ´etape," Ph.D. dissertation, Universit Blaise Pascal, GRIL, Clermont-Ferrand, 1998.
14. J. L. Paulo, "PAsMo – P´os-An´aliSe MOrfol´ogica," Laborat´orio de Sistemas de L´ıngua Falada (L^{2}F – INESC-ID), Lisboa, Relat´orio T´ecnico, 2001.
15. F. M. M. Batista, "An´alise sint´actica de superf´ıcie," Master's thesis, Instituto Superior T´ecnico, Universidade T´ecnica de Lisboa, July 2003.
16. P. Clarkson and R. Rosenfeld, "Statistical language modeling using the cmu-cambridge toolkit," in *Proc. Eurospeech '97*, Greece, Sept. 1997.
17. D. M. de Matos, J. L. Paulo, and N. J. Mamede, "Managing Linguistic Resources and Tools," in Lecture Notes in Artificial Inteligence, no. 2721, Springer-Verlag, 2003, pp. 135–142.
18. Massachusetts Institute of Technology (MIT), The MITRE Corporation. *Galaxy Communicator (DARPA Communicator)*. See: http://communicator.sf.net.
19. S. Seneff, E. Hurley, R. Lau, C. Pao, P. Schmid, and V. Zue, "Galaxy-ii: A reference architecture for conversational system development," in *Proc. ICSLP '98*, Sydney, Australia, Dec. 1998.

20. W3C, *The Extensible Stylesheet Language*, World Wide Web Consortium (W3C), 2001, see: www.w3.org/Style/XSL.
21. ECMA International, Geneva, Switzerland. *Standard ECMA-262 – ECMAScript Language Specification*, 3rd edition, December 1999. See also: www.ecma.ch.
22. PHP Group. *PHP Hypertext Processor*. See: www.php.net.
23. World Wide Web Consortium (W3C). *Extensible Markup Language (XML)*. See: www.w3.org/XML.
24. World Wide Web Consortium (W3C). *HyperText Markup Language (HTML)*. See: www.w3.org/MarkUp.
25. David M. de Matos, Alexandre Mateus, Jo˜ao Graça, and Nuno J. Mamede. Empowering the user: a data-oriented application-building framework. In *Adj. Proc. of the 7th ERCIM Workshop "User Interfaces for All"*, pages 37–44, Chantilly, France, October 2002. European Research Consortium for Informatics and Mathematics.