Tra-la-Lyrics: An approach to generate text based on rhythm

Hugo R. Gonçalo Oliveira
CISUC
Dep. of Informatics Engineering
University of Coimbra, Portugal
hroliv@dei.uc.pt

F. Amílcar Cardoso
CISUC
Dep. of Informatics Engineering
University of Coimbra, Portugal
amilcar@dei.uc.pt

Francisco C. Pereira
CISUC
Dep. of Informatics Engineering
University of Coimbra, Portugal
camara@dei.uc.pt

Abstract

This paper is about an ongoing project, *Tra-la-Lyrics* which aims to create a computer program capable of generating lyrics for given melodies. As a preliminary phase, the correlations between words and rhythm in a song's lyrics was studied and some heuristics were achieved. Algorithms for syllabic division and syllabic stress identification on a word were implemented. A method for calculating the strength of each beat of a melody was also used. The actual system's architecture is described. To get the contents of the lyrics we've used a relational database where we could find not only the words, but also their grammatical category and some morphological related attributes. Some ongoing work is also referred and some examples of the currently possible outputs of the system are presented. Conclusions and possible further work are finally discussed.

Keywords: computational creativity, rhythm, text generation, music, lyrics

1 Introduction

In the last few years we've been starting to accept the computer as a an artificial artist, highly contributing for turning the creative systems an interesting topic for research. There are many known creative systems with purposes like, for example, composing musical pieces (Cope (1987)), telling stories (Bringsjord and Ferrucci (1999); Gervás et al. (2006)), making up jokes (Binsted (1996)), creating visual art (Machado and Cardoso (2000)), making up poetry (Manurung (2004); Gervás (2000); Díaz-Agudo et al. (2002); Gervás (2001)) and so on. During these later years we have also seen a huge growth of systems where users can customize some digital and multimedia contents they can share with their whole community or simply keep them for their own use. Combining these two with the challenge it represents for research and the fun we hope it will provide, our objective is to design and develop a system capable of generating lyrics in the Portuguese language, from given melodies.

Lyrics have strict metrics that gives us the notion of rhythm. Metrics is obtained by the existence of syllables with different strengths. Also very important is the choice of words, which should not only obey the metrics, but also employ some phonetic patterns like the use of rhyme or alliterations. Although we may argue lyrics have always an emergent semantics, the semantic aspects are not a present concern of our project. We may even state that there shouldn't be a predefined semantics at all letting the users have their own free interpretation. Most of the times it is possible to make some interpretation out of any word sequence, specially if the words follow some gramatical structure.

With this system, we hope to contribute to the general endeavor of building artificial artists. It intends to accomplish one of the less explored tasks in automatic music production: writing the lyrics, given the melody. As far as we know, there were no previous attempts to develop such a system.

The structure of this paper starts by referring some important authors and applications in the area of creative text generation, as they can be included in the same category of the system are creating. After presenting some preliminary work, we will introduce the system itself, *Tra-la-Lyrics*. There will be given special focus to its actual architecture, the way it represents some important contents and also some ongoing work. At the end three sample generated lyrics will be shown.

2 State of the Art

2.1 Poetry generation systems

From the existing systems, poetry generating ones are probably the closest to ours. In this subsection we will refer some works in the area of poetry generation and their similarities and contributions to our system.

In his thesis, Hisar Manurung proposes an evolutionary approach for poetry generation and provides a computational model of metrical similarity, the minimum edit distance (Manurung (2004)). In this thesis there is a whole chapter dedicated to poetry and automatic poetry generation where he describes some interesting aspects like a definition for poetry, something about the process of creating poetry and a categorization for the existent poetry generation systems. He defines poetry in a way that it can be falsifiable and though a possible topic for research, experiment and evaluation. A poetic text should follow the properties of meaningfulness (M), grammaticality (G) and poeticness (P). It must convey some conceptual message, obey the given grammar rules and have some poetic features such as rhythm and rhyme. According to the properties they are in agreement with, he divides the existing poetry generation systems into four categories: word salad (respects no property); template and grammar-based (respects G property); form-aware (respects either P or G and P property); and poetry generation (respects G, P and M property). The minimum edit distance, used to evaluate the metrics, basically picks a candidate verse and compares its metrics with some target pattern. The distance is calculated by the number of syllables we would have to change so that the candidate verse fits the metrics.

Pablo Gervás has many works in the area. He was one of the first authors to make serious attempts to poetry generation. He created the WASP system (Gervás (2000)) whose main objective was to study the effect of the initial data in the resulting poem and to test different generating strategies. ASPERA (Gervás (2001)) and COLIBRI (Díaz-Agudo et al. (2002)) are examples of semi-automatic poetry generation system that take advantage of case-base reasoning techniques to generate Spanish poetry. In (Gervás (2001)) it is referred that there are three major challenges concerning automatic poetry generation, respectively the specification of the formal requirements that define a correct poem, the appropriate management of an extensive vocabulary and the correct combination of words.

2.2 Related applications

We can find many online applications able to generate creative text in the *web*. *The Poetry Creator* (Lewis and Sincoff (2006)) is a system based on verse templates, where the user is asked to give a subject, a synonym for the subject and a title for the poem. The generated text includes both the subject and its synonym in the middle of some poetic verses. There seems to be no present notion of metrics or rhymes. As an example we have generated a poem whose subject was *music* and the given synonym was *songs*. We got verses like *"Alms for the poor!" cried the quickly, ecstatic music With lightning strokes, the songs shot foreward;briefly, keenly*. In *The Essay Generator* (Mullen (2006)) generates an essay about some topic the user gives as an input. The generated essay includes not only words but also figures and references. They have always the same structure with an introduction respectively followed by the *Socials Factors*, *Economic Factors*, *Political Factors* and *Conclusions* sections. Nevertheless the essays are most of the times completely nonsense, because they merely include the topic in previously made sentences. As an example we have generated an essay on the topic ‘music and lyrics‘ and got an essay with sentences like *“Difference among people, race, culture and society is essential on the survival of our world, however music and lyrics smells of success.“* or *“The question which we must each ask ourselves is, will we allow music and lyrics to win our vote?”*. *SCIGen* (SCIgen (2005)) is an automatic computer science research paper generator, including graphs, figures and citations. The user only gets to include the name of the authors of the paper. The contents of the paper are then generated using a context-free grammar. *SCIGen*’s main objective is to test whether some conferences have low submission standards. The generated paper titled *Rooter: A Methodology for the Typical Unification of Access Points and Redundancy*[1] was once accepted.

Poesybeat (Initiative (2005)) is not a creative text output system, but an interesting collaborative system maintained by an online community. Users can do one of two things. Either they can send poems they would like to have a melody so they could be sung, or they can send melodies they wanted to have lyrics. The community users try then to find suitable choices. The final result is recorded so that everyone can listen to it. We are basically trying to automatize the *search of lyrics for a melody* part.

3 Preliminary study

This section briefly presents some preliminary work. We needed to study about the correlation about the words and the melody notes in the song lyrics.

3.1 Syllables

Common sense tells us that, in a song, each syllable of a word is associated to a note, so we had to find a way of dividing Portuguese words into syllables. In order to accomplish that, we’ve implemented our own algorithm. Although it is based in the algorithm presented in (Velloso (2006)), which was made for word wrapping in text processors, we had to add many unhandled situations. Our algorithm is based in some special groups of characters, shown in Figure 1.

The original algorithm divides the words into syllables using the groups VOWELS, CONJ1, CONJ2 and CONJ3. Each group of vowels identifies a different syllable. A complete syllable can also include consonants so, each consonant will be included in the previous or next syllable, depending on which of the groups (CONJ1, CONJ2, CONJ3) it belongs to. For example, for the word *reintegrar* we got the division *rein-te-grar*. We have added a second part for the algorithm, that uses the division made in the first part and then checks whether there is a syllabic division in the middle of a group of vowels. This only happens if the group is neither a dipthong nor part of a (’g’, ’q’) + ’u’ + vowel construction. For example. given the division *rein-te-grar*, we get the final division *re-in-te-grar*. In this word, the dipthong “ei” doesn’t exist because the letter ’i’ is followed by a nasal consonant.

3.2 Lyrics and rhythm

In order to generate lyrics for the given melodies we already knew there were some important rules, suggested

[1]http://pdos.csail.mit.edu/scigen/rooter.pdf

```
VOWELS = ('a', 'e', 'i', 'o', 'u', 'á', 'à', 'ã', 'â', ...);
SEMI_VOWELS = ('i', 'u');
GA = ('á', 'à', 'é', 'í', 'ó', 'ú');
CIRC = ('â', 'ê', 'î', 'ô', 'û');
TIL = ('ã', 'õ');
H = ('h');
U = ('u');
CONJ1 = ('b', 'c', 'd', 'f', 'g', 'p', 't', 'v');
CONJ2 = ('c', 'l', 'n');
CONJ3 = ('l', 'r', 'z');
CONJ4 = ('q', 'g');
```

Figure 1: Groups of characters used in syllabic division

in some online courses like the one from Berklee *Lyric Writing: Writing Lyrics to Music* (Pattison (2002)) and other online sources like (Simon (2006); Demeter (2001)). Perhaps the most important rule is to make sure that the stressed syllables are associated with the strong beats in a way that the lyrics have the melody's metric (Hayes and Kaun (1996)). In order to have some support for this rule and to find out some other correlations between the syllables of the lyrics and the beats of the melody we've implemented a simple information extraction program. The program would give us some information particularly for Portuguese. Its input was a set consisting of the melody and the lyrics of 42 Portuguese popular songs with the most usual types of meter[2] and its output was a set of tables where we had information related to the amount of stressed and unstressed syllables occurring in strong beats and also in weak beats. The information we obtained was in agreement with the rule of Bruce Hayes. Additionally, we collected information about the resolutions of weak syllables in strong beats, syllable prolongation and syllable contraction.

3.3 Stressed syllables and strong beats detection

To identify the stressed syllables in the words, we have implemented an algorithm that used the syllabic division and searched for characters and situations that could stress some syllable. Some examples of these situations are the presence of a graphical accent or the word terminating in a vowel followed by an *r*, *l* or *z*. There is also a list of 18 unstressed monossylables: *o*, *a*, *os*, *as*, *um*, *uns*, *me*, *te*, *se*, *lhe*, *nos*, *lhes*, *que*, *com*, *de*, *por*, *sem*, *sob*, *e*, *mas*, *nem*, *ou* (Sílaba (2006)). All the words that didn't match any of the rules were considered as being stressed in their penultimate syllable. This happens to be a rule for the Portuguese language.

We have used the dots system, present in *A Generative Theory for Tonal Music* (Lerdhal and Jackendoff (1983)) in order to get the strength of the beats present in the most usual types of meter. Considering level 0 the strongest, we obtained the results shown in Figure 2 in what concerns to the syllables present in the 42 songs. We divided each note/syllable pair into three categories: notes with stressed syllables, notes with unstressed syllables and linked notes. Linked notes are usually associated with a prolongation of the syllable in the previous note and are as strong as the beat in the note they are linked to, so we could consider them as neither stressed nor unstressed syllables.

[2]We have considered the following meter types: 2/4, 3/4, 4/4, 3/8 and 6/8

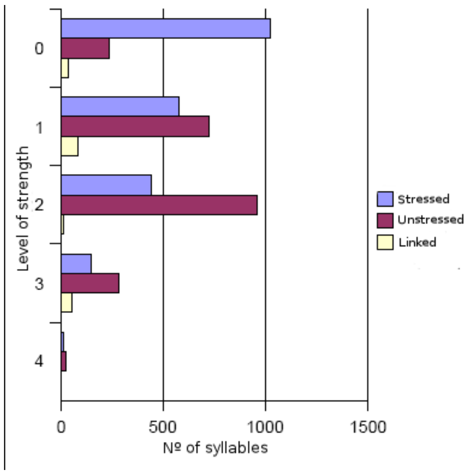


Figure 2: Distribution of syllables per level of strength

3.4 Rhymes and Repetition

As suggested in *How to Write Lyrics* (Demeter (2001)), there are three important points one should pay attention when choosing words for a song: **rhythm**, **rhyme** and **repetition**. We have already referred how we handle the rhythm. In order to provide some rhymes in the right places, we have added the possibility of giving the program a list with the notes where we want them to occur. Each melody can have its own list. Rhymes are get by placing words with the same termination in the given locations. In a later version we intend to automatize or assist the identification of these locations by using a melody segmentation analysis similar to the one in (Grilo (2002)). Repetition has a very tight relationship with rhythm and it is often present in visual art and music. When it comes to repetition, we have simply added a component that lets us control the probability of reusing previously chosen words, if they fit.

4 Tra-la-lyrics: System description

4.1 System architecture

Our system is implemented in the *Java* programming language and uses *MySQL* as a database engine. In Figure 3 it is possible to see the generic architecture of Tra-la-Lyrics' current version.

4.1.1 Inputs and Output

The basic input of our system is a MIDI file, which is converted into the ABC notation (Gonzato (2003)) using an external tool like *midi2abc*. After a couple of stages, suitable text for the given rhythm will be generated. At the end, we'll have a new ABC file, consisting of the original one with the generated lyrics inserted. Using tools like *abc2ps* and *ps2pdf* we can easily get PDF sheet music from the ABC file.

4.1.2 The modules

We briefly explain each one of the modules of our system:

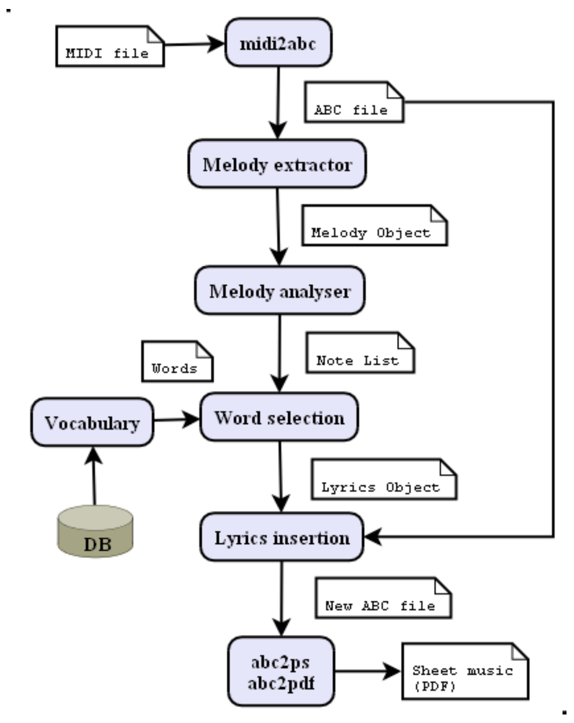
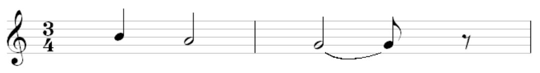


Figure 3: Generic system architecture



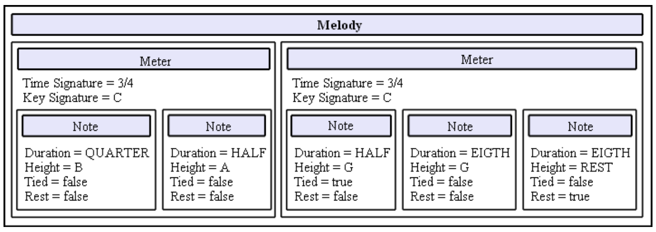


Figure 4: The *Melody* Object for the above sample notes

- **Melody extractor:** this module gets information from the ABC file, using the *ABC4J API* (Gueganton (2005)) and builds a *Melody Object* (Figure 4), which is basically a list of *Meters*. The *Meters* themselves are lists of *Notes*.

- **Melody analyzer:** for each *Note* of the melody, a new object *NoteInsideMeter* is created. It contains not only the *Note* object, but also its position inside the meter, its strength and whether it is a note corresponding to the end of a part and thus suitable to be the start of a rhyme[3]. We get the strength of the notes using the dots system present in *A Generative Theory for Tonal Music* (Lerdhal and Jackendoff (1983)). The output of the *Melody analyzer* is an object consisting of a list of *NoteInsideMeter* objects and some methods to get the distance (number of notes) between a given note and the next where a **special pattern** occurs. We call this object *Notes* and its rep-

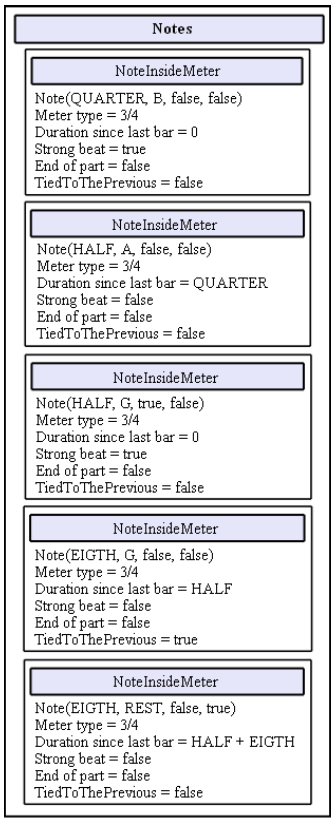
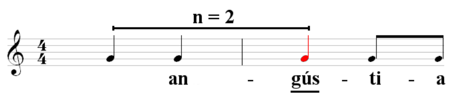

Notes
NoteInsideMeter
Note(QUARTER, B, false, false)
Meter type = 3/4
Duration since last bar = 0
Strong beat = true
End of part = false
TiedToThePrevious = false
NoteInsideMeter
Note(HALF, A, false, false)
Meter type = 3/4
Duration since last bar = QUARTER
Strong beat = false
End of part = false
TiedToThePrevious = false
NoteInsideMeter
Note(HALF, G, true, false)
Meter type = 3/4
Duration since last bar = 0
Strong beat = true
End of part = false
TiedToThePrevious = false
NoteInsideMeter
Note(EIGTH, G, false, false)
Meter type = 3/4
Duration since last bar = HALF
Strong beat = false
End of part = false
TiedToThePrevious = true
NoteInsideMeter
Note(EIGTH, REST, false, true)
Meter type = 3/4
Duration since last bar = HALF + EIGTH
Strong beat = false
End of part = false
TiedToThePrevious = false

Figure 5: The *NoteInsideMeter* Object

resentation for the melody in Figure 4 is shown in Figure 5. The **special patterns** will be listed shortly. Each one of them is an event sequence relevant for fitting the metrics of the text in the metrics of the melody.

- **Word selection:** This module is responsible to get words that fit in the melody rhythm. The *Notes* object is iterated and after calculating the **distance** from the current note to the next **special pattern** a word should be obtained from the *Vocabulary* module. If we get a new word, the next note's index will be needed. It is got by adding the number of syllables of the new word plus the number of rests and the number of linked notes found to the current index.
In the following we are listing all the considered **special patterns**. For each one of them, different kinds of words will be needed, depending on the calculated **distance** n. The words are supplied by the *Vocabulary* module.

  - **Strong beat**: A note occurs in a strong beat. We have marked as strong, the beats whose calculated strength is the highest of the meter or higher than some predefined value.



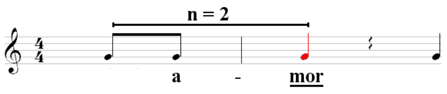
The word should have at most n+2 syllables[4]

[3]In the current version this info is provided by the user.

[4]Portuguese words are stressed in one of their last 3 syllables

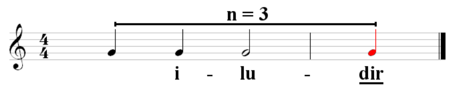
and its stress should be found in the *n*th syllable counting from the beginning of the word. For the example in the picture, we would need a word with 4 syllables with its second syllable stressed, like *an-**gús**-ti-a*.

– **Strong beat followed by a rest**: A note occurs in a strong beat and we can find a rest after it.



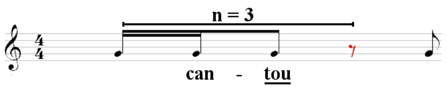
The word should have *n* syllables, and its stress should be found in the *n*th syllable, counting from the beginning of the word. For the example in the picture we would need a word with 2 syllables with its last syllable stressed, like *a-**mor***.

– **Strong beat followed by the end of the melody**: The note occurs in a strong beat and it's the last note of the melody.



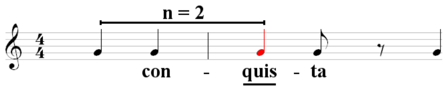
Similar to the previous pattern. For the example in the picture, we would need a word with 3 syllables, with its stress appearing in the last one, like *i-lu-**dir***.

– **Rest**: A rest.



The word should have at most *n-1* syllables, and should not have stress at all. We have only considered 18 unstressed words, all of them monosyllables, making it difficult to have long sequences of unstressed words. However it does not appear to be critical to have a few stressed words not in strong beats.

– **Strong beat followed by some note, followed by a rest**: The note is a strong beat, followed by some other note, after which we can find a rest

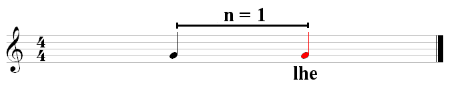


The word should have at most *n+1* syllables and its stress should be found in its *n*th syllable. For the example in the picture, we would need a word with at most 3 syllables, with its 2nd syllable stressed, like *con-**quis**-ta*.

– **Last strong beat of the current part**: The note is in the last strong beat of the current part of the music.

The word should have at most *n+d* syllables, where *d* is the distance between the last strong beat of the part and the last note of the part. The *n*th syllable must be the stressed one.

– **End of the melody**: The note is the last of the melody.



The word should be one of the 18 unstressed monosyllables, because this pattern means we have the last note of the melody and it's not in a stressed beat.

After having the lyrics for the whole sequence of notes, a *Lyrics Object* is built, consisting of a sequence of *Words*, which on their own are sequences of *Syllables*.

- **Vocabulary:** this module returns words, with a maximum number of syllables and a fixed distance to the stressed syllable. The words are obtained from a database, consisting of tagged words for the Portuguese language and some of their syllabic attributes like the syllabic division or the stress position. The method used to get the words can be reimplemented so that we can easily test different strategies.

- **Lyrics insertion:** at this stage the information in the original ABC file is merged with the generated lyrics, giving rise to a new ABC file.

4.1.3 The Database

As a source of words, we have used at first a database consisting of the words in the treebank *Floresta Sintá(c)tica*, which has a great amount of grammatically tagged sentences taken from the Portuguese newspaper *Público*. This treebank can be found in *Linguateca* (Linguateca (2000)). Due to some mistagged words and an unfriendly database structure (with some information we wouldn't need), we have created a new database with words taken from *Floresta Sint(c)tica*, tagged with a tool called *Jspell* (Jspell (1995); Dias de Almeida and Pinto (1995)). Everytime we want, we can use a simple program we have implemented to complement this database with words taken from other texts (poetry for instance). The new database keeps information about the grammatical categories of the words, their root, their gender, their number and also specific information about verbs or personal pronouns, like tense or person.

A table containing some syllabic related attributes for the words, like syllabic division, stressed syllable position or word termination was included in both databases. In order to get those attributes, the algorithms referred in Sections 3.1 and 3.3 were used.

4.2 Ongoing work

4.2.1 Strategies to get the words

As said before, the *Vocabulary* module has a method used to get words, based on both the position of their stressed syllable and the maximum number of syllables. This module keeps the strategy of getting words, making the *Words Selection* module simpler and the management and testing of different strategies easier. At the moment we have implemented three strategies:

- **Strategy 1, random words + rhymes:** where the *Vocabulary* simply returns random words and tries to have rhymes in the end of some given parts. The rhymes are simply based on the words' termination.
- **Strategy 2, words following sentence templates + rhymes**: where the *Vocabulary* module has a set of simple Portuguese sentence templates that are basically a sequence of grammar tags. The returned words are forced to follow some randomly chosen templates so that there is some syntactical coherence. There is also a submodule responsible for keeping the gender and the number of the last article or pronoun, forcing the following noun, adjective or verb to have the same morphology. With some probability two templates can be linked using a conjunction.
- **Strategy 3, grammar + rhymes** (currently in development): where the *Vocabulary* module uses a grammar whose derivations build Portuguese sentence templates. Each symbol in these templates is an instance of some grammatical category and has a set of attributes[5] depending on the grammatical category it belongs to. This strategy differs from the above one, specially in the following:
  - Backtracking is used when there are no suitable words to correctly finish a sentence;
  - If a list with the musical parts division is provided, it tries to make each sentence correspond to a musical part;
  - More grammatical categories are used and their attributes can easilly be defined;
  - Only open class words[6] are reused.
  - Each production has a different probability of being chosen so that most common sentences are more frequently built that less common ones;
  - It is able to generate a larger amount of different templates;

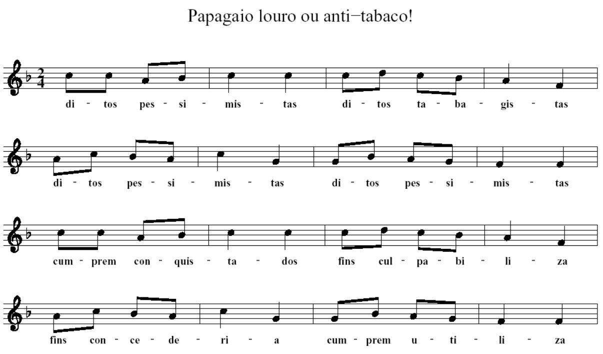
Both strategies 2 and 3 try to get words that rhyme in the end of some parts but, beyond the stress position and the number of syllables, theses strategies have to deal with restrictions like the category of the word and its morphological attributes, making it sometimes impossible to find words that rhyme. In order to minimize this problem, the program does the following: 1) Each time it needs to get a word for the end of a part to start a rhyme, it tries to choose between the matching words with the most common terminations. This maximizes the probability of finding a words that rhymes for the following verse(s); 2) Everytime we end a verse with a different termination, we store that termination. If there no words that follow all the restrictions and rhyme with the previous verse, we try to find words that rhyme with other terminations we had stored.

4.2.2 *Finding suitable locations for rhymes*

At the moment, it is possible to include in the ABC file a list with with some numbers corresponding to the notes where we'd like to have a rhyme. We are planning to include an evolutionary algorithm similar to the one used in (Grilo (2002)) so that the suitable places to have rhymes are automatically proposed.

[5]gender, number, person...

[6]nouns, adjectives and verbs.


Papagaio louro ou anti−tabaco!
di - tos pes - si - mis - tas di - tos ta - ba - gis - tas
di - tos pes - si - mis - tas di - tos pes - si - mis - tas
cum - prem con - quis - ta - dos fins cul - pa - bi - li - za
fins con - ce - de - ri - a cum - prem u - ti - li - za

```
Ditos pessimistas
ditos tabagistas
ditos pessimistas
ditos pessimistas
cumprem conquistados
fins culpabiliza
fins concederia
cumprem utiliza
```

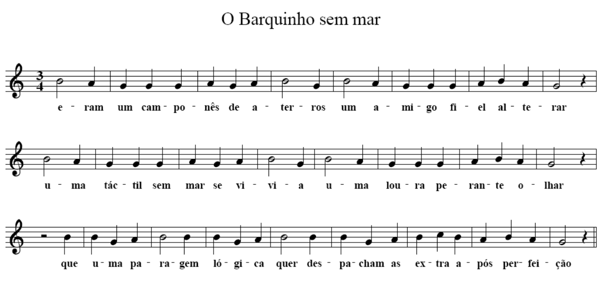
Figure 6: Sample generated lyrics using strategy 1

5 Demo runs

In this section we will ilustrate our system's behaviour with the analysis of three generated lyrics. Each one of them uses one of the previously referred strategies. The probability of reusing previously chosen words was set to 20% in the first and second and to 60% in the third. This probability is however only applied when the Vocabulary module is queried for a word and there are suitable words that have already been used in the same lyrics.

- The music shown in Figure 6 is a known Portuguese popular song called *Papagaio Louro*. The lyrics were generated by our program using **Strategy 1**. The lyrics are perfectly matching the rhythm and many words like *ditos* and *pessimistas* have been reused creating a somehow funny effect. We can find rhymes in the desired places (*tabagistas* and *pessimistas*, *culpabiliza* and *utiliza*). However, the words sequence doesn't obey grammatical rules.

- The music shown in Figure 7 is a known Portuguese popular song called *O Barquinho*. The lyrics were generated by our program using **Strategy 2**. Once again the lyrics match the rhythm, even though this strategy has more restrictions to get the words making the probability of not matching the rhythm a little higher. This example has also a rhyme in the right place (*alterar* and *olhar*). The most different aspect they have relative to the lyrics generated with **Strategy 1** is that the words follow some simple sentence templates. The lyrics we present follow templates like:


O Barquinho sem mar
e - ram um cam - po - nês de a - ter - ros um a - mi - go fi - el al - te - rar
u - ma tác - til sem mar se vi - vi - a u - ma lou - ra pe - ran - te o - lhar
que u - ma pa - ra - gem ló - gi - ca quer des - pa - cham as ex - tra a - pós per - fei - ção

```
Eram um camponês de aterros
um amigo fiel alterar
uma táctil sem mar se vivia
uma loira perante olhar
que uma paragem
lógica quer
despachem as extra
após perfeição
```

Figure 7: Sample generated lyrics using strategy 2

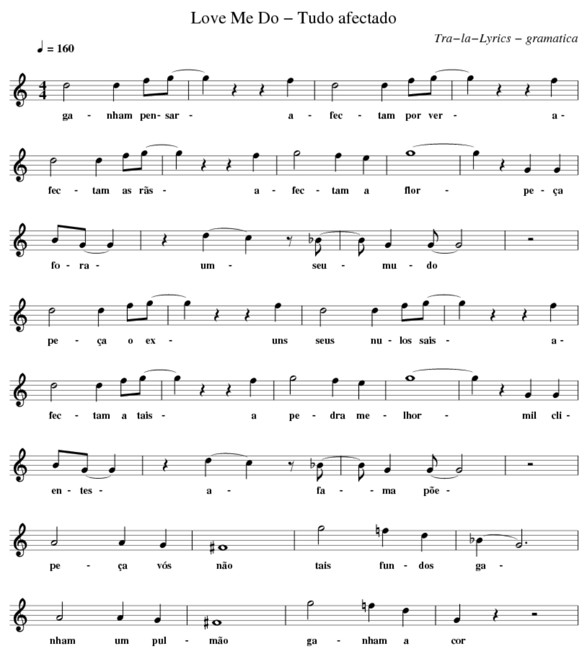
- verb article adjective preposition noun: *eram um camponês de aterros*
- article noun adjective: *um amigo fiel*

The probability of getting a conjunction between two templates was set to 80%, but the only present conjunction is the word *que*. Although we could get some meaning from the lyrics generated with both strategies, there is no explicit semantics. **Strategy 2** simply generates (almost) grammatically correct sentences.

- The music in Figure 8 is a well known song by *The Beatles*, *Love Me Do*. The shown lyrics were generated using **Strategy 3**. They match the rhythm and follow a large amount of templates, like:

  - verb preposition verb: *afectam por ver*
  - verb article noun: *afectam as rãs*
  - article possessive-pronoun adjective noun: *uns seus nulos sais*
  - article noun adjective: *a pedra melhor*
  - article noun verb: *o fama põe*

  This music is divided into many small parts and that's why the lyrics have only short sentences. We can find some rhymes like *sais* and *tais* or *não* and *pulmão*. We can also find the repetition of words like *peça*, *afectam* and *ganham*. The probability of reusing previously chosen words was set to a higher value (60%) because: 1) it is more difficult to find used words that match not only syllabic restrictions, but also category and morphology restrictions; 2) this strategy only tries to reuse words of open classes, as it is said Section 4.2.1. Like the lyrics generated with **Strategy 2**, these have no explicit semantics, however it is easier to build up some meaning out of them, because they follow grammatical rules.
  This strategy has still a wide range for improvement.


Love Me Do – Tudo afectado
Tra–la–Lyrics – gramatica
♩ = 160
ga - nham pen - sar - a - fec - tam por ver - a -
fec - tam as rãs - a - fec - tam a flor - pe - ça
fo - ra - um - seu - mu - do
pe - ça o ex - uns seus nu - los sais - a -
fec - tam a tais - a pe - dra me - lhor - mil cli -
en - tes - a - fa - ma põe -
pe - ça vós não tais fun - dos ga -
nham um pul - mão ga - nham a cor

```
Ganham pensar afectam por ver
afectam as rãs afectam a flor
peça fora um seu mudo
peça o ex uns seus nulos sais
afectam a tais a pedra melhor
mil clientes a fama põe
peça vós não tais fundos
ganham um pulmão ganham a cor
```

Figure 8: Sample generated lyrics using strategy 3

6 Conclusions and further work

Despite the results already achieved we hope to get better ones after refining some details and add new functionalities. We are planning to let the users change some settings, like the strategy used to generate the lyrics, the probability of reusing words or the places where they'd like to have rhymes. It would also be interesting to give them the possibility of having a set of preferred words or even texts, which would have more probability of being included in their generated lyrics.

In the future, more strategies for fitting the words in the rhythm or to get words can be tested. One different approach to fit the words could be similar to the one used by Hisar Manurung in (Manurung (2004)). We could analyze the melody and create its metric pattern that would be our target. We could then have a population of possible lyrics. To evaluate each lyrics in the population we could use the minimum edit distance, which would give us the amount of syllables not matching the target pattern.

Another strategy for obtaining the words will soon be tested, using the portuguese version of a (still) unpublished surface text realizer. Although the current version of the system only generates portuguese lyrics, it won't be difficult to change it to other languages. The syllabic division and syllabic stress detection algorithms would have to be reimplemented and a different source of words would be needed for each language.

As we all know, creative systems are not easy to evaluate. We are thinking in three different ways of evaluating our system. For testing if the words correctly fit in the rhythm, we can pick up a set of sample generated lyrics, use the information extraction system implemented during the preliminary work, and analyze the obtained information. We will be able to compare the syllabic distribution in our lyrics with the one in the "real" song lyrics. To evaluate the quality of the general output the better way would probably be to have a considerable amount of people answering questionaires, where they would be asked to compare existing lyrics with some generated, both for the same song. Another possible way of evaluating the lyrics' quality could be the implementation of some system that would analyze a set of generated lyrics for the same melody, and using some heuristics, evaluate the presence of poetic features and vocabulary diversity.

Another interesting thing we could do in a near future would be using some sung voice synthesis software, like Singing Computer (Zamazal (2001)), to sing our generated lyrics. Singing Computer uses Lilypond music notation files as input and there is one known ABC to Lilypond converter (abc2ly), making our task easier.

References

Binsted, K. (1996). *Machine humour: An implemented model of puns*. PhD thesis, University of Edinburgh.

Bringsjord, S. and Ferrucci, D. A. (1999). Artificial intelligence and literary creativity: Inside the mind of brutus, a storytelling machine. Lawrence Erlbaum Associates, Hillsdale, NJ.

Cope, D. (1987). An expert system for computer-assisted music composition. *Computer Music Journal 11,4 (Winter)*.

Demeter (2001). How to write lyrics. http://everything2.com/index.pl?node=How%20to%20write%20lyrics.

Dias de Almeida, J. J. a. and Pinto, U. (1995). Jspell - um módulo para a análise léxica genérica de linguagem natural. http://natura.di.uminho.pt/~jj/pln/jspell.ps.gz.

Díaz-Agudo, B., Gervás, P., and González-Calero, P. A. (2002). Poetry generation in colibri. In *ECCBR '02: Proceedings of the 6th European Conference on Advances in Case-Based Reasoning*, pages 73–102, London, UK. Springer-Verlag.

Gervás, P. (2000). Wasp: Evaluation of different strategies for the automatic generation of spanish verse. *Proceedings of the AISB00 Symposium on Creative and Cultural Aspects and Applications of AI and Cognitive Science*.

Gervás, P. (2001). An expert system for the composition of formal spanish poetry. *Journal of Knowledge-Based Systems*, 14:181–188.

Gervás, P., Lönneker-Rodman, B., Meister, J. C., and Peinado, F. (2006). Narrative models: Narratology meets artificial intelligence. In Basili, R. and Lenci, A., editors, *International Conference on Language Resources and Evaluation. Satellite Workshop: Toward Computational Models of Literary Analysis*, Genova, Italy.

Gonzato, G. (2003). The abc plus project. http://abcplus.sourceforge.net.

Grilo, C. F. A. (2002). *Aplicação de Algoritmos Evolucionários à Extracção de Padrões Musicais*. PhD thesis, University of Coimbra.

Gueganton, L. (2005). abc4j. http://gueganton.chez-alice.fr/abc.

Hayes, B. and Kaun, A. (1996). The role of phonological phrasing in sung and chanted verse. *The Linguistic Review*.

Initiative, P. A. (2005). poesybeat. http://poesybeat.org/.

Jspell (1995). Jspell. http://natura.di.uminho.pt/natura/natura?topic=jspell.

Lerdhal, F. and Jackendoff, R. (1983). *A Generative Theory of Tonal Music*. The Massachussets Institute of Tecnhology, 2nd edition - 1996 edition.

Lewis, J. and Sincoff, E. (2006). Poetry creator 2. http://www-cs-students.stanford.edu/~esincoff/poetry/jpoetry.html.

Linguateca (2000). Linguateca. http://www.linguateca.pt.

Machado, P. and Cardoso, A. (2000). Nevar - the assessment of an evolutionary art tool. In *Wiggins, G. (Ed.). Proceedings of the AISB00 Symposium on Creative & Cultural Aspects and Applications of AI & Cognitive Science, Birmingham, UK*.

Manurung, H. (2004). *An evolutionary algorithm approach to poetry generation*. PhD thesis, University of Edinburgh.

Mullen, D. (2006). Essay generator. http://radioworldwide.gospelcom.net/essaygenerator/.

Pattison, P. (2002). Lyric writing: Writing lyrics to music. http://www.berkleeshares.com/download/870472/berklee_musical_stress.pdf.

SCIgen (2005). Scigen - an automatic cs paper generator. http://pdos.csail.mit.edu/scigen/.

Sílaba (2006). A sílaba. http://www.geocities.com/shiurtalmid/catan/portuguese/v6.html.

Simon, T. (2006). Criterios para relacionar letra e musica. http://www.musicaeadoracao.com.br/tecnicos/musicalizacao/letra_musica.htm.

Velloso, A. T. (2006). Separando slabas com c#. http://www.microsoft.com/brasil/msdn/Tecnologias/visualc/SeparandoSilabas.mspx?mfr=true.

Zamazal, M. (2001). Singing computer. http://freebsoft.org/singing-computer.