Proceedings of the LREC 2020 Workshop Games and Natural Language Processing, pages 59–67
Language Resources and Evaluation Conference (LREC 2020), Marseille, 11–16 May 2020

Bringing Roguelikes to Visually-Impaired Players by Using NLP

Jesús Vilares, Carlos Gómez-Rodríguez, Luís Fernández-Núñez, Darío Penas, Jorge Viteri
Universidade da Coruña, CITIC, LyS Research Group, Dept. of Computer Science & Information Technologies
Facultade de Informática, Campus de Elviña, 15071 – A Coruña, Spain
{jesus.vilares,carlos.gomez,j.viteri.letamendia}@udc.es {fernandezn.luis,dariosoyyo}@gmail.com

Abstract

Although the roguelike video game genre has a large community of fans (both players and developers) and the graphic aspect of these games is usually given little relevance (ASCII-based graphics are not rare even today), their accessibility for blind players and other visually-impaired users remains a pending issue. In this document, we describe an initiative for the development of roguelikes adapted to visually-impaired players by using Natural Language Processing techniques, together with the first completed games resulting from it. These games were developed as Bachelor's and Master's theses. Our approach consists in integrating a multilingual module that, apart from the classic ASCII-based graphical interface, automatically generates text descriptions of what is happening within the game. The visually-impaired user can then read such descriptions by means of a screen reader. In these projects we seek expressivity and variety in the descriptions, so we can offer the users a fun roguelike experience that does not sacrifice any of the key characteristics that define the genre. Moreover, we intend to make these projects easy to extend to other languages, thus avoiding costly and complex solutions.

Keywords: Natural Language Generation, roguelikes, visually-impaired users

1. Introduction

One of the main factors for the huge growth of the video game industry has been the radical improvement of the graphic quality of video games. Paradoxically, this fact significantly hinders the access to these products by users with severe visual impairments such as blindness. Aware of this situation, some members of our Natural Language Processing (NLP) research group[1] decided to do our small part to help. Within this initiative, named TOP PLAYER LYS,[2] we have been offering to our students the chance of developing specially adapted roguelike games as their Bachelor's or Master's theses. These games should be accessible to both sighted and visually-impaired players by using NLP techniques (Jurafsky and Martin, 2009; Manning and Schütze, 1999). To do this, the games offer, apart from the standard game mode, a *descriptive mode* intended for visually-impaired users. In this game mode, the classic graphic representation of the dungeon and its elements is replaced by automatically generated natural language descriptions.

The choice of this particular genre is intentional and is due to several factors: the fact of being a game genre yet to adapt to this type of users; the existence of a large and well organized fan community, including both players and developers; and the relatively low value given to the graphic quality in these games. This allows us to ignore accessory aspects and focus on generating an accurate description.

This initiative of ours presents different aspects of interest, both at the social and academic levels. From a social point of view, we are not only concerned with how to extend the current offer of this type of entertainment to this sector of the population, but also with favoring their integration, since both sighted and visually-impaired users can play the same game and, thus, share common experiences. By helping draw students' attention to accessibility concerns, students will be aware of them when they go on to become software developers. Finally, from an academic point of view, this type of final-year projects allow us to introduce the student to NLP in a practical and engaging way. This is not a minor issue for us since no NLP course is currently available in our Computer Engineering Degree.

Regarding the rest of the paper, Section 2. describes the tools available to visually-impaired users and the relation of these users with video games. Next, Section 3. introduces the roguelike genre, its main features, and why this genre was chosen for this initiative. Section 4. makes a detailed analysis of the pre-requisites to be taken into account. Section 5., the core of this document, describes the games we have developed until now, their architecture, the linguistic levels involved and the solutions applied. Finally, Section 6. outlines our conclusions and future work.

2. Visually-Impaired Users

Tiflotechnology is a type of assistive technology that enables the practical use of high-tech devices to people who are blind or with low vision (Hersh and Johnson, 2008). Regarding computers, some of the proposed solutions involve the use of screen reader software (e.g. JAWS,[3] NVDA[4] and ORCA[5]), screen magnifiers (either software or hardware), OCR software and refreshable braille displays, among others. In the case of Web navigation, a careless design of a website may prevent screen readers from working properly on it. To avoid this, it is very important to follow basic accessibility guidelines, such as the WCAG 2 (W3C, 2018), to create a user-friendly website for blind people.

Another interesting issue is the fact that most software developers, even when designing accessible software, do not take into account that, when operating a computer, blind users often work in pairs with a sighted partner to help them in the event of a problem. Therefore, an accessible interface should make the same updated information available

[1] http://www.grupolys.org/

[2] http://www.grupolys.org/~jvilares/topplayerENG.html

[3] http://www.freedomscientific.com/Products/software/JAWS/

[4] http://www.nvaccess.org/

[5] http://wiki.gnome.org/Projects/Orca

to both the visually-impaired user and their sighted partner in every moment.

Concerning computer games, paradoxically, technological advances have made them less and less accessible to this sector of society. *Interactive text adventures*, for example, were a big part of the early days of gaming back in the 70s and part of the 80s (Barton, 2008). In this game genre, gameplay consists in reading brief text passages describing the current state of the game. In response, the player types a brief command describing the action to be taken by the main character of the story which, in turn, results in a new state. These input commands consist in simple phrase constructions, mainly directions and VERB+OBJECT structures. Their origin dates back to 1975, when Will Crowther, a professional programmer, designed and distributed the game *Colossal Cave Adventure* on the ARPANET, becoming a very popular title. By not containing graphics, except perhaps some static graphic as a supplement to the setting, they could be played by blind gamers by using text-to-speech synthesizers.

However, the progressive improvements of computers in terms of computing power and graphic capabilities, together with the availability of better and cheaper video terminals, made mainstream computer games progressively more focused on graphical aspects. As a result, games became less and less accessible for the blind, who were relegated to more mundane and simple games, such as board or card game adaptations. This led some developers to adapt certain computer games for the blind by modifying their sound effects and, subsequently, creating games based solely on audio. These so-called *audio games* focus on the possibilities of game immersion through audio. The appearance of 3D positional sound in the 2000s made it possible to somehow represent the position of elements in space through sound. This allowed audio game developers to approach new genres (Urbanek and Güldenpfennig, 2019).

3. The Roguelike Genre

The origin of this genre is in the computer game *Rogue*, created by two enthusiasts, Michael Toy and Glenn Wichman, in the early 80s (Craddock, 2015). Originally developed for UNIX mainframes, it became very popular after being included with BSD UNIX, thus making it virtually available at universities all over the world and giving birth to the *Rogue*-like genre.

Influenced by sword-and-sorcery tabletop role-playing games, in *Rogue* the player controls a hero who explores a dungeon complex. As the player's character defeats lurking monsters, avoids traps and discovers treasures, they will be awarded with *experience points*. As the character gains experience, they become stronger, so they can face greater challenges. The dungeon also hides a multitude of items such as weapons, armors, amulets, etc., which may give extra bonuses or penalize the actions of the adventurer.

Among the main features of this first game and, by extension, of the genre, we remark the following:

- *Random environment generation.* The dungeon and its elements (enemies, traps, items, etc.) are randomly generated for every new game by using procedural content generation algorithms (Shaker et al., 2016), thus favoring replayability.
- *Random outcomes.* As with tabletop games, the calculation of the result of tactical actions, such as attacks or spells, includes a random component to add certain degree of variability and tension.
- *Permanent consequences.* Saves are only permitted between gameplay sessions and are automatically deleted after loading. This also implies that the player's character has a single life (*permadeath*).
- *Turn-based.* Each command corresponds to a single action which, in turn, takes a single turn with no time limit. This allows the player to take their time to assess the situation and decide what to do.
- *Grid-based.* The dungeon is represented by a uniform grid of orthogonal tiles, where every element (adventurer, monsters, items, etc.) takes up a single tile. Movement between tiles is atomic.
- *ASCII (pseudo-)graphics.* Although modern roguelikes, mainly commercial ones, usually have a true graphical display, classic roguelikes were played on text terminals using a text-based display instead. As depicted in Figure 1,[6] the dungeon map and its elements are drawn from a top-down view by using ASCII characters. This allows to reduce the computer requirements of the game and favors portability. Even nowadays, many roguelike fans prefer to play them, if possible, using ASCII graphics: a good roguelike should be judged according to its game mechanics and the player experience, not by its graphics.

Figure 1: Gameplay screenshot of a *Rogue* UNIX clone. The dungeon map is drawn using '−' and '|' for room walls, '.' for room floors, '+' for doors, '#' for passageways between rooms, and '%' for stairs connecting dungeon levels. Regarding other characters and elements, the player's character (the hero/adventurer) is represented as an '@', an enemy monster (a jackal) as a 'J', gold coins as '⋆' and weapons as ')'.

[6] Obtained from *Retro Rogue Collection*: https://github.com/mikeyk730/Rogue-Collection

Figure 2: Gameplay screenshot of the game *Dungeon Crawl Stone Soup*, where we can see the complexity of these text-based interfaces.

Nowadays, these features may not be impressive to many current gamers but, back then, they were innovative and have had a great influence in subsequent games of all genres (Craddock, 2015).

Today, roguelike games count on a large group of enthusiastic fans, both players and developers, either amateur or professional. They are well-organized around webs such as *RogueBasin*[7] and *Temple of the Roguelike*,[8] where they share their experiences. At the commercial level, apart from the influence they had (and still have) in other genres, roguelikes are in good shape, particularly among indie developers. In the last years, the genre has even given rise to a new subgenre, the *roguelite* or *roguelike-like* games: a less strict interpretation of the genre that tries to bring it closer to the general public (Craddock, 2015).

3.1. Why Roguelikes

There are several reasons for having chosen the roguelike genre for our initiative of developing games adapted to visually-impaired players by using NLP techniques.

The very first motive was that, even having wide experience as both players and amateur developers of these games, as far as we knew no specifically adapted roguelike was available. Furthermore, our previous contacts with blind gamers supported that assumption. As one of them, who was a fan of the genre, explained to us, the only way he could play these games was by using a Braille display. This means that, every turn, he had to read the screen contents, line by line, using the gadget. In parallel, he had to form in his mind some kind of mental representation of the map, its elements and his stats, so he could make a decision about what to do next. As can be guessed by looking at Figure 2, corresponding to the popular roguelike *Dungeon Crawl Stone Soup (DCSS)*,[9] the time and mental effort needed for such a task must be noteworthy. It is not easy for a blind player to discern (and then remember) the required information about the current state of the game in such a jumble.

The second reason for our choice was the existence of an active and well-organized community of players and developers. This fact guaranteed the availability of freely-available resources (such as tutorials, code libraries, discussion forums, etc.) which should reduce the effort of implementing the core game itself, allowing us to focus on our NLP problem. Moreover, with such a wide and dedicated fan community, the chances of our work to be improved or extended to new languages increased.

Finally, as explained above, many roguelike players pay little attention to the game graphics, instead focusing on the core playing experience. This means that we do not need to worry about creating state-of-the-art graphics and, again, can mainly focus on the language generation issues.

4. Pre-Requisites

Our goal was to develop roguelike games specially adapted to visually-impaired players as Final-Year Projects (i.e., Bachelor's and Master's Theses) of a Bachelor's and Master's degree in Computer Engineering. These games should provide visually-impaired players, using natural language, with an accurate description of the dungeon and its elements, in such a way that they can assess what action to take next. This *descriptive mode* would require the use of NLP techniques (Jurafsky and Martin, 2009; Manning and Schütze, 1999). In the case of sighted players, they can keep playing the game using the standard ASCII-based *graphic mode*. However, we should also take into account a major limitation: that the curriculum of said degrees includes no NLP courses. Therefore, before launching our idea, it was necessary to establish clear prerequisites that were compatible with this context:

- It should be conceived as an introduction to NLP, since this field is not covered by the curriculum.
- The difficulty and scope of the project must fit the workload assigned to these projects: 12 ECTS credits (300 hours).
- Therefore, the main effort should focus on the development of the description generator module. The game itself should be simple, as it acts only as a "demonstration platform". Issues such as graphic quality, variety of enemies and items, creature AI, etc., should be secondary.
- The system must be designed with *multilingualism* in mind. Thus, it should be able to generate descriptions in at least two languages: English and Spanish. The reason for this is twofold. The number of potential English-speaking users is much higher, as is the availability of NLP tools and resources. On the other hand, Spanish, apart from being the native language for our students and hence an easy starting language, is also the first language of an important number of potential users and, after consulting in specialized forums, we could confirm that a good portion of them have problems with English. Furthermore, with a more complex morphology and syntax than English, it can provide an idea of the scalability of our solutions.
- At the same time, the system should be flexible and easily extensible and modifiable by third parties.

[7] http://www.roguebasin.com
[8] https://blog.roguetemple.com/
[9] https://crawl.develz.org/

Thus, they could improve aspects of the basic game or add new languages to the description module.

- To do this, we should avoid complex solutions. Although there is a wide community of amateur developers, we know nothing about their competence level. Our intention is that any person with basic programming skills and high-school linguistic skills should be able to extend the system, at least in part, to a new language.

- Additionally, our experience with low-resource languages suggested that the number and complexity of the linguistic resources to be used should be low. Therefore, they should be simple to obtain or build, if needed. Note that the free availability of these types of resources is often limited (Rehm and Uszkoreit, 2011).

- The collaboration of beta-testers is mandatory in a project like this, including not only sighted players but, above all, visually-impaired ones. Their perspectives are different but complementary, and reliable feedback from both sides was needed. The help obtained from specialized forums and from trainers from the National Organization of Spanish Blind People,[10] was invaluable.

- The game should allow either blind or sighted users to play the same game. This has a triple effect: (1) it makes debugging much easier; (2) when working in pairs, as described in Section 2., it makes the work of the sighted partner easier; and (3) if the same game can be played by either visually-impaired or sighted users, they will be sharing the same experience (a very similar one, at least), thus favoring integration.

- Special attention should be paid to *usage licenses*. The games should be made freely available in the Web under a non-commercial open source license. This means that third-party resources used must be compatible with such a license.

5. Game Development

Until today, we have been developed three games with different approaches to the problem. All of them are freely available under a GNU General Public License v3 at our website:

1. *The Inner Eye*, by Luis Fernández-Núñez;

2. *The Accessible Dungeon*, by Darío Penas;

3. and *Hamsun's Amulet*, by Jorge Viteri.

As an example, Figure 3 shows one screenshot of *The Accessible Dungeon*. In what follows, instead of describing them separately, we will analyze the different solutions adopted in their design and implementation together.

[10]Organización Nacional de Ciegos Españoles (ONCE): `https://www.once.es/`

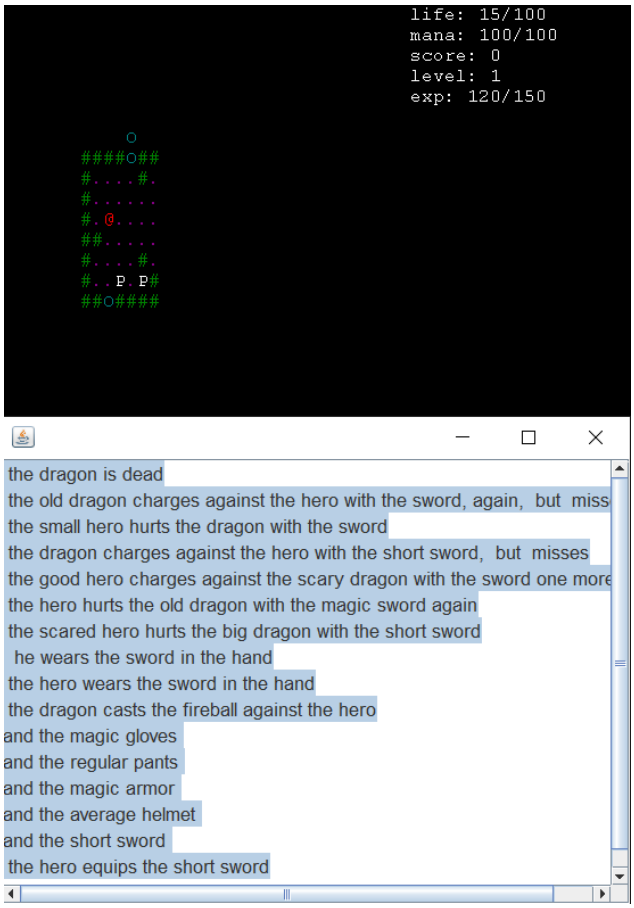

Figure 3: Gameplay screenshot of *The Accessible Dungeon* with its double display: the classic ASCII graphics (top) and the output of the description module (bottom). Note than, following a suggestion of our beta-testers, the content of this later text window must be read in reverse order (from bottom to top).

5.1. System Architecture

Figure 4 shows the general architecture of our games. The main difference with respect to regular roguelikes is the addition of a so-named *Text Description Engine* module. This new component takes as input the (current) world model and, together with the information provided by the game logic in response to the commands of the user, it generates a textual description of the dungeon and what is happening within the game. Such description is generated according to the linguistic resources available and the language selected. Regarding this new module, it follows the classic architecture of a Natural Language Generation (NLG) system. It is composed of three stages (Reiter and Dale, 2000):

- *Content planning*. Determines the content and structure of the description.

- *Microplanning*. Selects the words and syntactic structure to be used to express such content.

- *Surface realisation*. Integrates all this information and transforms the abstract representation of the message into actual text to be presented to the user.

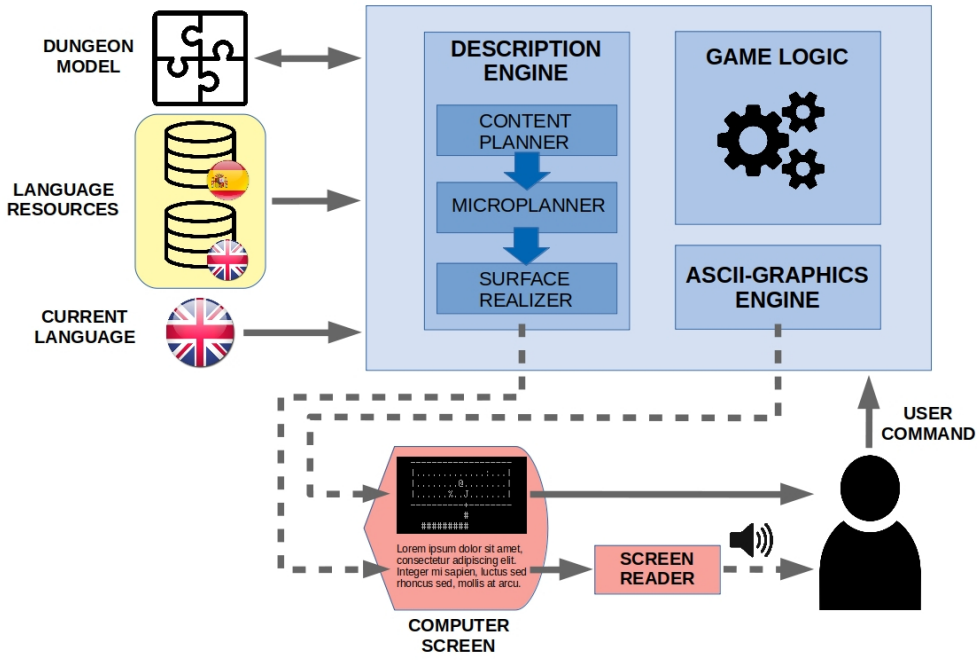

Figure 4: General architecture of the system.

Especial attention has been paid to the microplanner, seeking expressivity and variety. For this purpose, we intend to take advantage of the so-called *linguistic variation* (a.k.a. *linguistic flexibility*): the ability of our languages to express the same message in very different ways (and vice versa) (Arampatzis et al., 2000).

5.2. Linguistic Processing

The various solutions adopted cover different levels of linguistic processing, from the lexical to the pragmatic level, without losing sight of multilingualism. At this point, the languages that have been addressed are: English, Spanish, Galician and Dutch. Preliminary work has been made for Japanese, too.

5.2.1. Lexical and Morphological Levels

Firstly, it is necessary to rely on a *system vocabulary* that includes every element, action and situation that may occur or appear within the game. The terms of this vocabulary also need to be complemented with their corresponding morphosyntactic information; for example, its word category, its Part-Of-Speech (POS) tag, etc.

Each of these terms must be conveniently indexed with an external identifier. This key must be conveniently referenced in the corresponding program entities or processes with which it is involved. For instance, if the user gives the order to attack a goblin, the text generator receives from the program logic the information that the *player's character* (`ID_hero`) is performing an *attack action* (`ID_toAttack`) in which the target is a *goblin* (`ID_goblin`) and the instrument used is a sword (`ID_sword`) —the weapon they currently have equipped. In our particular case, the use of the English word lemmas as identifiers yielded good results, being both meaningful and manageable. Once obtained, the generator module can now retrieve the actual terms corresponding to those key identifiers in the currently-selected language. This procedure is very similar to the way in which assets are organized and managed during game localization (Chandler and Deming, 2011), and the way in which synsets from different languages corresponding to the same meaning are interlinked through the Interlingual Index (ILI) in the case of WordNet-like lexical resources (McCrae and Cimiano, 2015).

Finally, the system also needs to be able to obtain the inflectional variants of a term due to changes in gender, number, person, etc. To do this, one option was the use of inflectional generators (Jurafsky and Martin, 2009, Ch. 3). Taking as input the word lemma (e.g. "glorioso", Spanish for "glorious") and the required inflection (e.g. the feminine plural form), a tool of this type would generate the corresponding inflected term ("gloriosas"). However, we decided not to use these tools, at least initially. Otherwise, this would imply the need to either find an inflectional generator meeting our criteria (i.e. freely available for each of the given languages) or to build one from scratch. Both cases were problematic if we wanted to allow third parties to extend the system to new languages easily. On the one hand, this resource may not be available for every language; on the other hand, building a tool of this type from scratch requires NLP knowledge that an amateur would not have. Thus, we decided to adopt a simpler, albeit somewhat cumbersome, solution: to hold in the dictionary not only the lemma of the word, but also its inflectional variants.

These dictionaries have been implemented as JSON files because of their simplicity and flexibility. They are just text files with a structured and human-readable format, so they can be easily extended and modified even with a simple text editor. As an example, we show below the entry corresponding to the adjective "glorious" in the English dictionary of the game *The Accessible Dungeon*:

```
{
 "ADJ": {
    ...
    "glorious":
    [
       {"num": "sing"},
       {"translation": "glorious"},
       {"numopposite": "glorious"},
       {"gender": ""}
    ],
    ...
}
```

as well as the entries corresponding to its translation in the Spanish dictionary,[11] also indexed by their common key identifier (i.e. the original lemma in English):

```
{
 "ADJ": {
    ...
    "glorious":
    [
       {"num": "sing"},
       {"translation": "glorioso"},
       {"numopposite": "gloriosos"},
       {"genopposite": "gloriosa"},
       {"gen": "mas"}
    ],
    "gloriosa":
    [
       {"num": "sing"},
       {"translation": "gloriosa"},
       {"numopposite": "gloriosas"},
       {"genopposite": "glorious"},
       {"gen": "fem"}
    ],
    "gloriosos":
    [
        {"num": "plural"},
        {"translation": "gloriosos"},
        {"numopposite": "gloriosas"},
        {"genopposite": "gloriosas"},
        {"gen": "mas"}
    ],
    "gloriosas":
    [
        {"num": "plural"},
        {"translation": "gloriosas"},
        {"numopposite": "gloriosa"},
        {"genopposite": "gloriosos"},
        {"gen": "fem"}
    ],
    ...
}
```

As can be seen, it is a redundant format where, for each term, we have available its inflectional features and, for each of them, the term obtained by varying only that single feature with respect to the current one (e.g. masculine vs. feminine for gender, and singular vs. plural for number). This scheme allowed a simple navigation through the different inflected forms of the term when applying agreement restrictions such as, for example, the gender-number agreement between a noun and its adjectives in Spanish.
These dictionaries were created manually taking the English one as a reference, since it contains the minimum vocabulary to cover all the elements currently in the game.

In contrast, in the case of the game *Hamsun's Amulet*, a semi-automatic approach was chosen instead. The content word lemmas (nouns, verbs, adjectives and adverbs) of the basic game vocabulary still have to be selected manually. This first manual phase is mandatory in any game, since it is dependent on the design and implementation of the roguelike itself: the game developers are the ones that decide which elements, situations and actions are implemented and how they are related. However, given a lemma, its inflected forms were automatically extracted from an external corpus of the language.[12] Any annotated text corpus containing the POS tag and lemma of its terms would be suitable for the task. In this case, the corpus used was Wikicorpus (Reese et al., 2010), a freely available trilingual corpus (English, Spanish and Catalan) that contains large portions of a 2006 Wikipedia dump. The texts forming this corpus, over 750 million words, were automatically enriched with linguistic information including POS tags and lemmas. The format chosen for the resulting dictionaries, one for each language and word category, was also simpler and similar to the classic *(word, POS tag, lemma)* lexicon format, as we show here:

```
{
    ...
      "glorioso":{
         "type": "qualificative",
         "sing_m": "glorioso",
         "plu_m": "gloriosos",
         "plu_f": "gloriosas",
         "sing_f": "gloriosa"
      },
    ...
}
```

5.2.2. Syntactic Level

Until now, we have described how the game vocabulary is managed by the system. In this way, the description module is able to retrieve those terms corresponding to the elements and actions involved in a given gameplay event. However, these individual words have yet to be arranged to form a meaningful message describing said event. For that purpose, a *generation grammar* has been developed for every available language. In turn, this main grammar has been structured into *subgrammars* according to the syntactic structures generated (e.g. noun phrases) and the different contexts in which they are used: combat, use of magic, movement, generic actions, etc. This makes their management and maintenance easier. For example, if the developer intends to make a complete upgrade to the game's magic system, the possible changes to be made in the grammar will be restricted to that subgrammar.
As in the case of individual terms, the various subgrammars of the system and their rules are identified using an external identifier or key. This key is used to link each subgrammar with the events it describes, independently of the specific language being used at a given time.
These grammars are defined using context-free rules and kept in the form of JSON files. The notation employed for its specification, properly adapted to JSON format, is inspired by the one employed to specify the set of restrictions in feature structure-based grammars (Carpenter, 1992). We show as an example the definition of a simple noun phrase structure for English in the game *The Accessible Dungeon*:

```
"GENERAL": {
    "GM_3": {
        "S":
            [
            {"DET_1": ""},
            {"ADJ_1": ""},
```

[11] Translated as "glorioso", "gloriosa", "gloriosos" or "gloriosas" depending on the gender and number of the modified noun.

[12] Those inflected forms not appearing in the corpus, a minority, still had to be completed manually.

```
                {"N_1": ""}
            ],
            "restrictions": [
                {"DET_1.num": "N_1.num"},
                {"N_1.num": "ADJ_1.num"}
            ]
        }
    }
```

GENERAL is the key identifier corresponding to the current subgrammar and GM_3 is a rule identifier for internal use. We can distinguish two sections: S (for the classic start symbol), that describes the structure (i.e. the "right-hand side" of the rule); and restrictions, where we specify the morphosyntactic restrictions applicable to the elements of the structure. In this case, the grammatical structure corresponds to a sequence formed by a single DETerminer (DET_1), followed by an ADJective (ADJ_1) that modifies a Noun (N_1) (proper or common), as in the case of "a red potion", for example.

Regarding the restrictions of the example above, they specify that the number (feature num) of the determiner must be the same as that of the noun, and that the number of the noun must be the same as that of the adjective. In other words, we are informing the generator module about the well-known *number agreement* of a noun with its determiners and modifiers.[13] Thus, we will be avoiding the generation of ungrammatical phrases such as "a red potions".

Given a game event to be described, the generator engine will choose a random rule to be used among those of the subgrammar corresponding to that situation. By doing this, the generator module is taking advantage of *syntactic flexibility*, that is, using different syntactic structures to express the same message in a different way (Ferreira, 1996). Again, we are favoring variety and expressivity. For example, for the same attack action, we might obtain a brief description such as "The hero attacks the dragon", or an epic version such as "The mighty hero attacks the fierce dragon with his sword". So, by extending the grammars and/or subgrammars, any user can improve the quality and variety of descriptions.

5.2.3. Semantic Level

On this matter, it is interesting to take a look to the approach taken in *Hamsun's Amulet*. In this game, the terms forming the system vocabulary were not selected and organized individually, but at a *synset* level instead. For this purpose, its creator used the *Multilingual Central Repository (MCR) 3.0* (González-Agirre et al., 2012) as source. The MCR is a freely available WordNet-like resource that integrates in the same framework WordNets from six different languages: English, Spanish, Catalan, Basque, Galician and Portuguese. It also contains an Inter-Lingual-Index (ILI) that connects words in one language (actually synsets) with their equivalent translations (again, a synset) in any of the other languages.

Therefore, in this case the system vocabulary is composed not of a list of words corresponding to the different elements, actions and situations in the game, but of a list of synonym sets. As an example, and continuing with the one previously used in Section 5.2.1., the entries corresponding to the English adjective "glorious" and the rest of terms of its synonym set are these:

```
{
    ...
   "glorious":[
      "brilliant",
      "glorious",
      "magnificent",
      "splendid"
   ],
    ...
}
```

while their corresponding entries in the Spanish dictionary ("glorioso" and its synonyms) are:

```
{
    ...
   "glorious":{
      "glorioso":{
         "type": "qualificative",
         "sing_m": "glorioso",
         "plu_m": "gloriosos",
         "plu_f": "gloriosas",
         "sing_f": "gloriosa"
      },
      "brillante":{
         "type":"qualificative",
         "sing_m":"brillante",
         "plu_m":"brillantes",
         "plu_f":"brillantes",
         "sing_f":"brillante"
      },
      "magnifico":{
            ...
      },
      "esplendido":{
            ...
      }
   },
    ...
}
```

Again, an external identifier has been used to identify and link the entries between languages. This time, one of the lemmas of the original English synset is used as key. We could have just employed ILI-based keys but, in this context, the lemmas proved again to be more flexible and manageable with respect to a meaningless alphanumeric code.

The basic generation process of this synset-based game vocabulary did not differ too much from the regular one. After defining an initial word-level English vocabulary, its corresponding synsets were identified by means of an automatic matching at the lemma level followed by a manual revision to filter out incorrect senses. The use of Word-Sense Disambiguation techniques was immediately discarded because of its high cost and complexity, which were incompatible with our requirements (see Section 4.). Once the synsets to be used were delimited, their inflected forms were obtained as described in Section 5.2.1..

With respect to the description generation process, the only difference with respect to the regular one is that the specific term to be used at a given moment is selected randomly among those in the synset.

5.2.4. Discourse and Pragmatic Levels

Given the time and complexity restrictions associated to these projects, previously described in Section 4., little progress were made at these levels.

One of the features integrated consisted in changing the adjectives used to describe a character taking into account their current state, thus introducing a subjective point of

[13] Noun-adjective agreement is unnecessary in English (as adjectives don't inflect for number), but essential in languages like Spanish.

view. For example, if the number of hit points of an enemy creature is very low with respect to those of the adventurer, the generator module will reflect this fact by describing that enemy as "small", "insignificant", etc. If the situation is the opposite, adjectives such as "huge", "powerful", etc. will be used instead. Therefore, the enemy will be seen differently depending on the context. This mechanism gives the system more expressivity, while also improving the player experience by enhancing his empathy for the character.
To take into account persistence over time is another way of improving the player experience. For example, after having defeated an enemy, we have taken it into account for subsequent descriptions. Thus, for example, when the adventurer crosses again a room where they have fought and killed a goblin, the description should reflect this by automatically making reference to the dead body of the defeated enemy.

5.3. Other Relevant Features

Some extra accessibility-related features were added according to the comments and suggestions of our beta-testers and other members of the community.
The first one was the possibility of selecting how certain aspects of the game are described: in a qualitative or a quantitative way. For instance, in the case of hit points and other player statistics, some players preferred to be given an exact numeric value, but others preferred the use of fuzzy terms such as "high", "low", "enough", etc. A similar suggestion was made to describe the current position of the adventurer within a room, for example. Some players required the use of X-Y coordinates, while others were happy with approximate descriptions with respect to the elements of the room.
Another one was the configuration of other aspects of the interface, such as to enable the change of font size, the reassignment of command keys or the use of different color palettes in the case of color-blind users.
Finally, we were also asked to make a sound every time the player moves within the game, since sometimes this is the only feedback the player has about whether the action has been performed or not.

6. Conclusions and Future Work

Throughout this paper we have described the development of a description generator module to adapt roguelike games to visually-impaired users. This NLP-based module describes, in the form of text, what is happening within the game, enabling a blind person to play the game using a screen reader. Expressivity and variety are achieved by taking advantage of linguistic flexibility at diferent levels. With regard to design and implementation, our premises were simplicity, flexibility and extensibility, so that, once made available to the public, any user with basic knowledge of programming and linguistics could extend the game to other languages. Several languages have been successfully tried until now: English, Spanish, Galician and Dutch.
With respect to the future, now that we have achieved a certain amount of critical mass, we can extend this initial work in several possible ways, specially at the discourse and pragmatic levels. The management of temporal aspects such as the elimination of unnecessary repetitions with respect to recent events or considering other aspects of persistence, would be of interest. Another possibility is the addition of a "summary mode" that, taking as input the sequence of events that happened during the gameplay and their corresponding descriptions, could generate as output a story about the adventures of the player. However, the possible need for applying *storytelling* (Salen-Tekinbas and Zimmerman, 2004) and *narrative modeling* (Mani, 2012) techniques may be too much of a challenge given our present context. Other less obvious aspects of linguistic variation (Pérez L. de Heredia and de Higes Andino, 2019) such as to modify the tone of the description according to the psychological condition of the adventurer (e.g. injured and hungry vs. victorious and satisfied) or their career (e.g. a rude barbarian vs. a cultured wizard), for example, would also be worthy of time.
There are also other interesting features to add, but not so directly related to NLP. For example, improving and making the game configuration mechanisms more flexible, allowing individual features to be activated and deactivated. In the case of the descriptive mode for visually-impaired users, we are considering to split it in two: a *verbose mode*, with more detailed and extensive descriptions; and a *brief mode*, with minimalist descriptions for experienced players who want to streamline the gameplay. Finally, we also intend to improve the evaluation process through questionnaires for users.

Acknowledgements

The work of Profs. Vilares and Gómez-Rodríguez was partially funded by MINECO (through project ANSWER-ASAP TIN2017-85160-C2-1-R) and Xunta de Galicia (through grants ED431B 2017/01, ED431G/01 and ED431G 2019/01).
We also want to thank the reviewers for their helpful comments. Finally, we also want to express our gratitude to the roguelike and blind gaming community, specially to members of *Tiglo-juegos* Google group, and to Juan Carlos Buño-Suárez, trainer from the National Organization of Spanish Blind People (ONCE), for their help in sharing their needs, suggesting improvements and betatesting.

7. Bibliographical References

Arampatzis, A., van der Weide, T. P., van Bommel, P., and Koster, C. (2000). Linguistically-motivated information retrieval. In *Encyclopedia of Library and Information Science*, volume 69, pages 201–222. Marcel Dekker, Inc, New York-Basel.

Barton, M. (2008). *Dungeons and Desktops: The History of Computer Role-Playing Games*. A K Peters, Ltd, Wellesley, MA, USA.

Carpenter, B. (1992). *The Logic of Typed Feature Structures*. Number 32 in Cambridge Tracts in Theoretical Computer Science. Cambridge University Press, Cambridge/New York/Melbourne.

Chandler, H. M. and Deming, S. O. (2011). *The Game Localization Handbook*. Foundations of Game Development. Jones & Bartlett Learning, USA, 2nd edition.

Craddock, D. L. (2015). *Dungeon Hacks: How Nethack, Angband, and other Roguelikes Changed the Course of video Games*. Press Start Press, Canton, OH, USA.

Ferreira, V. S. (1996). Is it better to give than to donate? syntactic flexibility in language production. *Journal of memory and language*, 35(5):724–755.

Marion A. Hersh et al., editors. (2008). *Assistive Technology for Visually Impaired and Blind People*. Springer, London.

Jurafsky, D. and Martin, J. H. (2009). *Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition (2nd ed.)*. Pearson–Prentice Hall, Upper Saddle River, New Jersey, 2 edition.

Mani, I. (2012). *Computational Modeling of Narrative*. Synthesis Lectures on Human Language Technologies. Morgan & Claypool Publishers.

Manning, C. D. and Schütze, H. (1999). *Foundations of Statistical Natural Language Processing*. The MIT Press, Cambridge (Massachusetts) and London (England).

McCrae, J. P. and Cimiano, P. (2015). Guidelines for linguistic linked data generation: Wordnets. Draft Community Group Report 29, Best Practices for Multilingual Linked Open Data (BPMLOD) Community Group, World Wide Web Consortium (W3C). Available online at: `https://www.w3.org/community/bpmlod/` (visited on Feb. 2020).

Pérez L. de Heredia, M. and de Higes Andino, I. (2019). Multilingualism and identities: New portrayals, new challenges. In María Pérez L. de Heredia et al., editors, *Special Issue on Multilingualism and Representation of Identities in Audiovisual Texts*, volume 4 of *MonTI: Monografías de Traducción e Interpretación*, pages 9–31. Available online at: `http://hdl.handle.net/10045/96908` (visited on Feb 2020).

Georg Rehm et al., editors. (2011). META-NET White Paper Series. Springer. Available online at `http://www.meta-net.eu/whitepapers`.

Reiter, E. and Dale, R. (2000). *Building Natural Language Generation Systems*. Cambridge University Press, New York, NY, USA.

Salen-Tekinbas, K. and Zimmerman, E., (2004). *Rules of Play: Game Design Fundamentals*, chapter Games as Narrative Play. MIT Press.

Shaker, N., Liapis, A., Togelius, J., Lopes, R., and Bidarra, R. (2016). Constructive generation methods for dungeons and levels. In Noor Shaker, et al., editors, *Procedural Content Generation in Games: A Textbook and an Overview of Current Research*, chapter 3, pages 31–55. Springer.

Urbanek, M. and Güldenpfennig, F. (2019). Celebrating 20 years of computer-based audio gaming. In *Proceedings of the 14th International Audio Mostly Conference: A Journey in Sound*, AM'19, page 90–97, New York, NY, USA. Association for Computing Machinery.

W3C. (2018). Web content accessibility guidelines. Available at: `https://www.w3.org/WAI/standards-guidelines/wcag/` (visited on February 2020).

8. Language Resource References

González-Agirre, A., Laparra, E., and Rigau, G. (2012). Multilingual central repository version 3.0: upgrading a very large lexical knowledge base. In Christiane Fellbaum et al., editors, *Proceedings of the Sixth Global WordNet Conference (GWC 2012). Matsue, Japan*. Tribun EU. Resource available at: `https://adimen.si.ehu.es/web/MCR` (visited on Feb. 2020).

Reese, S., Boleda, G., Cuadros, M., Padró, L., and Rigau, G. (2010). Wikicorpus: A word-sense disambiguated multilingual Wikipedia corpus. In *Proceedings of the Seventh International Conference on Language Resources and Evaluation (LREC'10)*, Valletta, Malta, May. European Language Resources Association (ELRA). Resource available at: `https://www.cs.upc.edu/~nlp/wikicorpus/` (visited on Feb. 2020).