On the Design of an Artificial Player for a Popular Word Game*

Alberto Coffrini[1], Stefania Monica[2], and Federico Bergenti[1]

[1] Dipartimento di Scienze Matematiche, Fisiche e Informatiche
Università degli Studi di Parma, Italy
alberto.coffrini@studenti.unipr.it, federico.bergenti@unipr.it
[2] Dipartimento di Scienze e Metodi dell'Ingegneria
Università degli Studi di Modena e Reggio Emilia, Italy
stefania.monica@unimore.it

Abstract. This paper describes the design of an implemented software system intended to accomplish a specific natural language processing task. The targeted task is a challenge of the Evaluation Campaign of Natural Language Processing and Speech Tools for Italian proposed in 2020. The challenge is to design and implement an artificial player for the closing game of a popular Italian television show. Given five words, the goal of the player is to find a word related to, but also different from, the given words. The design of the proposed artificial player is discussed by presenting the dataset used to acquire sufficient linguistic knowledge and by briefly describing the algorithm used to play the game. A preliminary experimental evaluation of the artificial player is also discussed.

Keywords: Word games · Lexical semantics · Natural language processing · Artificial intelligence

1 Introduction

Natural Language Processing (*NLP*) is an interdisciplinary research field that overlaps computer science, artificial intelligence, and linguistics. The grand goal of NLP is to make computers able to process and understand human languages (e.g., [8]). Despite such an ambitious goal, current research on NLP targets restricted and specific goals to study relevant characteristics of natural languages or to design and implement solutions for specific tasks.

Logic programming has been playing a relevant role in NLP since the very first studies related to computational linguistics (e.g., [5]). Actually, the logic programming paradigm shares several characteristics with the approaches used to study natural languages. For example, the surface structure of a natural language is often described in terms of a lexicon and a set of grammatical rules. Hence, logic programs are particularly well suited to describe the surface structures of natural languages.

* Copyright © 2021 for this paper by its authors. Use permitted under Creative Commons License Attribution 4.0 International (CC BY 4.0).

As a general-purpose logic programming language, Prolog is normally considered an appropriate option to deal with NLP problems, and it has been effectively used to develop complex NLP systems (e.g., [3]) since its initial presentation. The integrated search and unification mechanisms that are intrinsic to Prolog make it an ideal candidate to implement formal models of languages. In addition, the declarative nature of Prolog allows focusing on problems rather than on their solving algorithms. Finally, a Prolog program is capable of conveniently supporting incremental knowledge, and such an ability is crucial when designing software systems that need to deal with open and dynamic contexts, such as those related to complex NLP problems.

It is worth noting that inductive logic programming (e.g., [9]) and probabilistic logic programming (e.g., [10, 11]) have also been successfully used to design and implement NLP systems. More recently, several approaches based on neural networks (e.g., [2]) and deep learning (e.g., [13]) were extensively used to target practical NLP problems. Finally, probabilistic methods and techniques have been intensively studied to target complex NLP problems (e.g., [4]).

The first studies on NLP date back to the 1950s, but the interest on the topic is still high and rapidly increasing because of the large assortment of possible application contexts for NLP. Traditional application contexts for NLP include machine translation, text classification, and question answering. In recent years, the advent of mobile technologies and the massive use of social networking services led to the rise of new application contexts, such as sentiment analysis for brand monitoring or for social media monitoring.

The accurate analysis of the targeted application context is crucial for the design and implementation of effective NLP systems. Actually, it is common opinion that different NLP tasks can be best accomplished using different approaches. A method to tackle a task typically achieves better performance in that task than in any other task, even if the tasks are closely related.

The software system discussed in this paper is specifically designed to accomplish an NLP task related to a popular word game called *La Ghigliottina* (Italian for *The Guillotine*), which is the closing game of one of the most popular Italian television shows. Given five words, the goal of the player is to find a sixth word related to, but also different from, the given five words. Possible relationships between the sixth word and each one of the given words include: being synonyms, being antonyms, or being used in the lyrics of a popular song. The goal of the NLP system discussed in this paper is to play this game in the attempt to find the sixth word related to a given set of five words. The challenge to design and implement artificial players for this game was included as a task called *Ghigliottin-AI* in the *Evaluation Campaign of Natural Language Processing and Speech Tools for Italian* (EVALITA) [1] in 2020.

This paper is organized as follows. Section 2 describes the procedure followed to collect and process the texts used to acquire the needed linguistic knowledge. Section 3 outlines the design of the proposed artificial player. Section 4 shows and comments some experimental results. Section 5 concludes the paper and discusses planned developments of the artificial player.

2 Construction of the Dataset

The first step to implement the artificial player for the considered word game regarded the preparation of a dataset to acquire sufficient linguistic knowledge. The dataset contains texts that were properly cleaned to remove punctuation marks and words that are not used in the game, such as articles and prepositions. After the initial cleaning phase, a set of pairs of words was constructed by considering words that are close to each other in the cleaned texts. For each pair of words, the number of occurrences in the cleaned texts was counted. The collected pairs of words and their number of occurrences are used to play the game, as briefly discussed in Section 3. The remaining of this section describes the initial cleaning phase and the construction of the set of pairs of words.

2.1 The Set of Cleaned Texts

The initial cleaning phase is intended to remove, from all considered texts, punctuation marks and words that are not useful to play the game. To this aim, a set of stop words was defined. The considered stop words are Italian words that are not used in the game, and they are mainly articles and prepositions. All occurrences of stop words were removed by parsing input texts to generate cleaned texts. For each input text, a new cleaned text is produced after the initial cleaning phase. The processing of input texts to remove stop words was also used to replace all punctuation marks with an uncommon symbol, namely $, that is conventionally used to break sentences. Note that, in order to speed up the initial cleaning phase, all input texts were split into separate text files with sizes limited to 1 MB.

The considered word game uses Italian words only, and therefore several texts written in Italian were collected to acquire sufficient linguistic knowledge. Various types of texts were chosen from different sources to ensure that the words required to play the game were actually contained in the considered texts. Note that some texts were written by Italian authors, while others are professional translations from other languages.

Thirteen freely available books were downloaded from e-book platforms and cleaned. They are mainly narrative books, and they include *Pinocchio*, *Zeno's Conscience*, *Don Quixote*, *Alice's Adventures in Wonderland*, and *The Little Prince*. A part of *Corpus Paisà* [7] was downloaded and cleaned. *Corpus Paisà* is a large collection of texts written in Italian that is commonly used to acquire linguistic knowledge on the Italian language. The considered part of *Corpus Paisà* corresponds to approximately 0.5 GB of text files.

The titles of all articles included in the Italian edition of Wikipedia were downloaded and cleaned. In addition, 150 complete articles included in the Italian edition of Wikipedia were downloaded and cleaned. The complete articles concern various topics, such as cooking, music, and travels. The articles are all about general knowledge topics because the considered game makes no assumptions on the education level of players.

Several texts containing lists of common compound words were downloaded and cleaned. In Italian, a compound word is a word that is composed of two other words, and the meaning of the compound word is not immediately related to the meanings of the two constituents. For example, *sempreverde* (Italian for *evergreen*) is a compound word composed of the two words *sempre* (Italian for *ever*) and *verde* (Italian for *green*). Note that a compound word is a regular word, and its two constituents are normally regular words. Therefore, besides considering the compound word *sempreverde*, the two words *sempre* and *verde* can also be considered as two separate words. It is worth noting that the division of compound words into theirs constituents is very useful to play the considered game. Actually, it is common that one of the constituents of a compound word is included in the five words given to the player, and the sixth word that the player should guess is the second constituent of the compound word. The compound word often represents the only relationships between the two words, and the player is expected to take compound words into account to play the game. Other examples of compound words in Italian are: *girasole* (Italian for *sunflower*), that is composed of the two words *gira* (Italian for *turn*) and *sole* (Italian for *sun*); *lanciafiamme* (Italian for *flamethrower*), that is composed of the two words *lancia* (Italian for *throw*) and *fiamme* (Italian for *flames*).

In addition to compound words, the considered texts also include lists of idiomatic phrases. Idiomatic phrases are phrases composed of few words that exhibit a specific semantic cohesion. Idiomatic phrases can belong to various lexical categories, such as nouns, adjectives, and adverbs. For instance, *gatto delle nevi* (Italian for *snowcat*) is used to indicate a vehicle designed to move on snow. This idiomatic phrase sets up a very unusual relationship between the word *gatto* (Italian for *cat*) and the word *nevi* (Italian for *snow*), and such a relationship is very useful to play the considered game. It is common that one of the words of an idiomatic phrase is included in the five words given to the player, and the sixth word that the player should guess is another word of the idiomatic phrase. Other examples of idiomatic phrases in Italian are: *di male in peggio* (Italian for *from bad to worse*); *meglio tardi che mai* (Italian for *better late than never*); and *nervi a fior di pelle* (Italian for *nerves on edge*).

Finally, the considered texts also include a list of Italian proverbs, which are often used informally and can be very useful to play the considered game. Examples of proverbs include: *l'erba del vicino è sempre più verde* (Italian for *the grass is always greener on the other side of the fence*); *non piangere sul latte versato* (Italian for *do not cry over spilled milk*); and *non dire gatto se non ce l'hai nel sacco* (Italian for *don't count your chickens before they hatch*). Proverbs create relationships among words that are not found in other contexts. Therefore, the inclusion of a list of proverbs is useful when playing the considered game to take into account common sentences of the everyday language.

Note that the inclusion of compound words, idiomatic phrases, and proverbs is very helpful to play the considered game. Preliminary empirical observations suggest that, on average, at least one of the five words given to the player is related to a compound word, an idiomatic phrase, or a proverb.

2.2 The Set of Pairs of Words

The player of the considered game is expected to find relationships among words. Actually, the player is expected to find a relationship between the guessed word and each one of the given words. For this reason, the texts obtained after the initial cleaning phase are further processed to obtain a set of pairs of words. Such pairs of words are obtained by parsing all cleaned texts and by considering two consecutive words in the same sentence as a pair.

In the remaining of this paper, the first word of each pair is called *token* and the second word is called *related token*. Using this nomenclature, a generic pair of words is denoted as

$$\langle\, token,\ related\ token\, \rangle. \tag{1}$$

For each pair of words, the *occurrence* of the pair corresponds to the number of times in which the pair is found in the cleaned texts. Given a pair of words, its *inverse* is the pair of words obtained by swapping the token with the related token. The occurrence of the inverse of a pair equals the occurrence of the original pair. The algorithm to play the considered game outlined in Section 3 uses the available pairs of words and their occurrences. Each one of the given five words is considered as a token, and the related tokens obtained by the available pairs of words are considered good guesses for the sixth word. The actual answer is chosen randomly among the related tokens with maximum occurrence.

The following example should clarify the roles of tokens, related tokens, and occurrences. Assume that the phrase *parcheggiare la macchina* (Italian for *park the car*) is found in a text. During the cleaning process, the article (namely, *la*) is removed, so that the two words *parcheggiare* and *macchina* are close to each other in the cleaned text. Then, the pair

$$\langle\, parcheggiare,\ macchina\, \rangle \tag{2}$$

is created and added to the dataset, where *parcheggiare* is considered as token and *macchina* is considered as related token. Assuming that this pair of words is found 5 times in the cleaned texts, the occurrence of the pair is equal to 5. Since the two words should be considered as related regardless of which is the token and which is the related token, the inverse pair

$$\langle\, macchina,\ parcheggiare\, \rangle \tag{3}$$

is also created and added to the dataset. The occurrence of the inverse pair is equal to the occurrence of the direct pair, which is 5. The direct pair ensures that, if *parcheggiare* is given, then *macchina* is a valid guess for the sixth word. Similarly, the inverse pair guarantees that, if *macchina* is given, then *parcheggiare* is a good guess for the sixth word.

All pairs of words obtained from the cleaned texts described at the beginning of this section were persistently stored together with their respective occurrences to provide the needed dataset to the artificial player. The current dataset contains more than 34,000 tokens, and each token is matched with a number of related tokens between 100 and 1,000.

3 The Implemented Algorithm in Brief

This section outlines the algorithm that is currently implemented in the artificial player for the considered word game. The algorithm works on a set of five words, and it returns the chosen guess for the sixth word. A preliminary experimental evaluation of the performance of this algorithm is discussed in Section 4.

The algorithm can be briefly summarized as follows. Each one of the five words is treated as a token of a word pair and it is searched in the dataset obtained using the method discussed in the previous section. To simplify the notation, it is assumed that all five tokens are found in the dataset. The set of five tokens is denoted as $\{t_i\}_{i=1}^5$, and, for each token t_i, with $1 \leq i \leq 5$, the set of its related tokens is denoted as R_i. All related tokens of the five tokens are considered as valid guesses for the sixth word. Actually, the sixth word is searched in the following set

$$R = \bigcup_{i=1}^{5} R_i. \tag{4}$$

In order to choose the returned sixth word among the related tokens in R, each related token in R is associated with two numeric values, called *frequency* and *match*. The values of frequency and match of each related token in R are used to choose the sixth word.

In order to properly define the frequency, let r_j be a generic element of R. If the pair $\langle\, t_i,\, r_j \,\rangle$ is found in the dataset, then it is associated with $o_{i,j}$, which equals the occurrence of the pair. Otherwise, the pair is conventionally associated with $o_{i,j} = 0$. The frequency of the generic related token r_j, denoted as f_j, is evaluated as the sum of the occurrences

$$f_j = \sum_{i=1}^{5} o_{i,j}. \tag{5}$$

Possible values for the frequency of a generic related token r_j are positive integer numbers. Note that the frequency f_j increases as the occurrences $\{o_{i,j}\}_{i=1}^5$ increase. Hence, the chosen sixth word is expected to have a large frequency.

The match of the generic related token r_j of R, denoted as m_j, is the number of tokens for which r_j is a related token. In other words, the match of the generic related token r_j is computed as the count of the pairs of words $\langle\, t_i,\, r_j \,\rangle$ that found in the dataset. Possible values for the match of a generic related token r_j are integer numbers from 1 to 5, where $m_j = 1$ if r_j is the related token of only one of the given words, and $m_j = 5$ if r_j is the related token of all given words.

Frequency and match are evaluated for each related token r_j in R, and they are used to decide which related token should be selected as sixth word. First, the set of guesses for the sixth word is restricted to the related tokens with the largest match. Then, the sixth word is chosen among the related tokens with the largest frequency. If two ore more related tokens share the same frequency, then the sixth word is chosen at random among them.

The following example should explain the roles of the frequency and the match of a related token, and it should clarify the approach behind the algorithm used by the player. Given a set of five words $\{t_i\}_{i=1}^{5}$, consider a simple dataset composed of the following pairs of words

$$\begin{array}{lllll} \langle\, t_1,\, r_1 \,\rangle, & \langle\, t_1,\, r_2 \,\rangle, & \langle\, t_1,\, r_3 \,\rangle, & \langle\, t_1,\, r_4 \,\rangle & \\ \langle\, t_2,\, r_1 \,\rangle, & \langle\, t_2,\, r_3 \,\rangle, & \langle\, t_2,\, r_4 \,\rangle & & \\ \langle\, t_3,\, r_1 \,\rangle, & \langle\, t_3,\, r_2 \,\rangle, & \langle\, t_3,\, r_4 \,\rangle & & \\ \langle\, t_4,\, r_1 \,\rangle, & \langle\, t_4,\, r_3 \,\rangle, & \langle\, t_4,\, r_4 \,\rangle, & \langle\, t_4,\, r_5 \,\rangle & \\ \langle\, t_5,\, r_1 \,\rangle, & \langle\, t_5,\, r_4 \,\rangle, & \langle\, t_5,\, r_6 \,\rangle, & \langle\, t_5,\, r_7 \,\rangle, & \langle\, t_5,\, r_8 \,\rangle \end{array} \tag{6}$$

Note that only the pairs of words related to the input words are listed in order to simplify the discussion of the example. Normally, a useful dataset contains a lot of other pairs of words.

For each pair of words $\langle t_i, r_j \rangle$ in the dataset, with $1 \leq i \leq 5$ and $1 \leq j \leq 8$, assume that its occurrence $o_{i,j}$ has the following values

$$\begin{array}{lllll} o_{1,1} = 2, & o_{1,2} = 1, & o_{1,3} = 2, & o_{1,4} = 1 & \\ o_{2,1} = 1, & o_{2,3} = 2, & o_{2,4} = 2 & & \\ o_{3,1} = 5, & o_{3,2} = 4, & o_{3,4} = 8 & & \\ o_{4,1} = 3, & o_{4,3} = 3, & o_{4,4} = 7, & o_{4,5} = 1 & \\ o_{5,1} = 7, & o_{5,4} = 8, & o_{5,6} = 5, & o_{5,7} = 1, & o_{5,8} = 9 \end{array} \tag{7}$$

Consider now, for example, the related token r_1. The frequency f_1 of the related token r_1 is

$$f_1 = \sum_{i=1}^{5} o_{i,1} = 18. \tag{8}$$

Since r_1 is a related token of all five tokens, the match m_1 of r_1 equals 5. Then, consider the related token r_3, whose frequency f_3 is

$$f_3 = \sum_{i=1}^{5} o_{i,3} = 7, \tag{9}$$

where $o_{3,3}$ and $o_{5,3}$ equal 0 because r_3 is neither a related token of t_3 nor of t_5. Since r_3 is a related token of three tokens, the match m_3 of r_3 equals 3. Finally, consider the related token r_8. Since r_8 is a related token of t_5 only, the frequency f_8 of r_8 equals $o_{5,8} = 9$. For the same reason, the match m_8 of r_8 equals 1.

Among the eight related tokens considered in the example, r_1 and r_4 are those with the largest match. Actually, r_1 and r_4 are related tokens of each one of the five tokens $\{t_i\}_{i=1}^{5}$, so that $m_1 = m_4 = 5$. Since the frequency of r_1 is $f_1 = 18$ and the frequency of r_4 is $f_4 = 26$, the sixth word guessed by a player that uses the discussed algorithm is r_4.

Note that, in order to simplify the notation, the description of the algorithm summarized in this section assumes that all input words are actually found in the dataset. However, rare input words may not appear in the dataset in real situations. In this case, the proposed algorithm does not change significantly, and the rare words are simply removed from the set of input words.

4 Preliminary Experimental Results

In order to preliminary assess the performance of the algorithm briefly discussed in the previous section, 100 instances of the considered game were played by the implemented artificial player. The considered instances were taken from the instances that were played in the television show. The success rate of the current version of the artificial player over the considered instances was 24%.

The following is an example of an instance of the game in which the player found the correct sixth word. The given five words are:

– *Chimico* (Italian for *chemical*)
– *Polizia* (Italian for *police*)
– *Segreto* (Italian for *secret*)
– *Cambio* (Italian for *change*)
– *Commercio* (Italian for *commerce*)

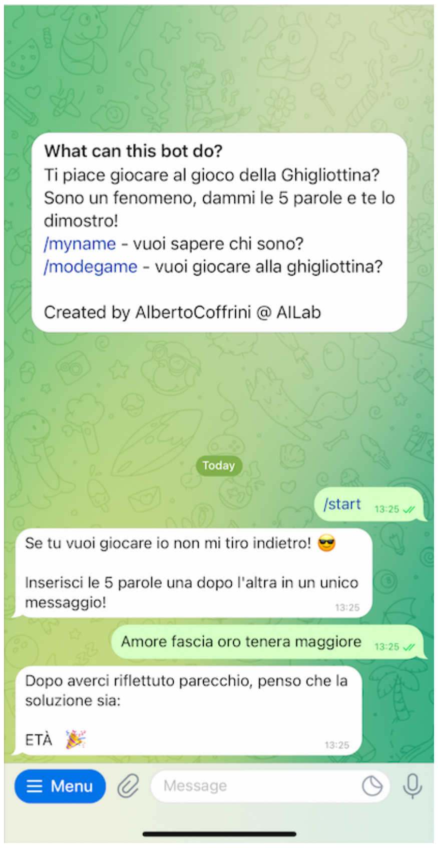
The correct sixth word, as found by the artificial player, is *agente* (Italian for *agent*). Actually, one can say in Italian: *agente chimico* (Italian for *chemical agent*); *agente di polizia* (Italian for *police officer*); *agente segreto* (Italian for *secret agent*); *agente di cambio* (Italian for *stockbroker*); and *agente di commercio* (Italian for *sales agent*).

The following is an example of an instance of the game in which the player did not find the correct sixth word. The given five words are:

– *Male* (Italian for *evil*)
– *Perdere* (Italian for *lose*)
– *Ordine* (Italian for *order*)
– *Campo* (Italian for *field*)
– *Natura* (Italian for *Nature*)

In this case, the correct sixth word is *forza* (Italian for *force*). Actually, in Italian one can say: *forza del male* (Italian for *force of evil*); *perdere forza* (Italian for *lose strength*); *forza dell'ordine* (Italian for *law enforcement*); *forza in campo* (Italian for *ground force*); and *forza della natura* (Italian for *force of nature*). Instead, the sixth word proposed by the artificial player was *gioco* (Italian for *game*). Although the proposed sixth word is wrong, it is worth noting that it is strongly related to some of the five words. Actually, one can say in Italian: *perdere un gioco* (Italian for *loose a game*); and *campo di gioco* (Italian for *playing field*).

Currently, the artificial player can be interfaced using a Telegram chatbot that was specifically created for the purpose. The user sends the message `/start` to the chatbot to start a new session. Then, the user can send to the chatbot commands or lists of five words. Supported commands are: `/myname`, to have the chatbot reply with a short greeting message; and `/modegame` to have the chatbot reply with a short message explaining how to play the game. Messages that do not start with commands are interpreted as lists of five words. When a list of five words is received, the chatbot queries the artificial player and replies to the user with the sixth word chosen by the artificial player. Normally, the artificial player chooses the sixth word in less than a second.


What can this bot do?
Ti piace giocare al gioco della Ghigliottina?
Sono un fenomeno, dammi le 5 parole e te lo dimostro!
/myname - vuoi sapere chi sono?
/modegame - vuoi giocare alla ghigliottina?

Created by AlbertoCoffrini @ AILab

Today

/start 13:25

Se tu vuoi giocare io non mi tiro indietro! 😎

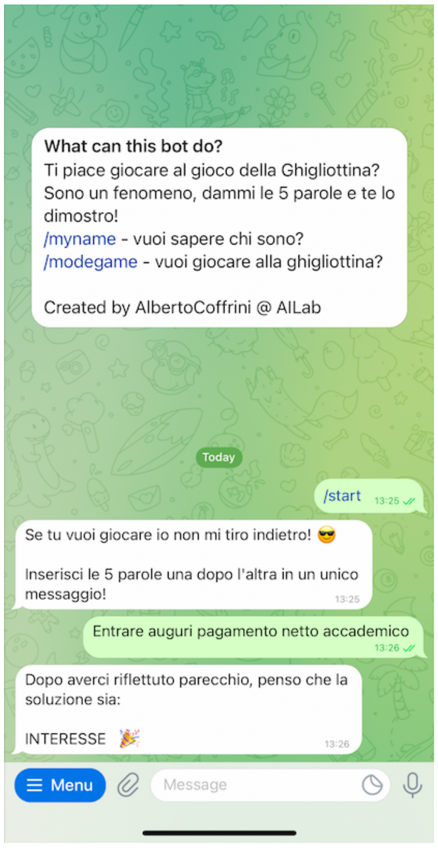
Inserisci le 5 parole una dopo l'altra in un unico messaggio! 13:25

Amore fascia oro tenera maggiore 13:25

Dopo averci riflettuto parecchio, penso che la soluzione sia:

ETÀ 🎉 13:25

(a)


What can this bot do?
Ti piace giocare al gioco della Ghigliottina?
Sono un fenomeno, dammi le 5 parole e te lo dimostro!
/myname - vuoi sapere chi sono?
/modegame - vuoi giocare alla ghigliottina?

Created by AlbertoCoffrini @ AILab

Today

/start 13:25

Se tu vuoi giocare io non mi tiro indietro! 😎

Inserisci le 5 parole una dopo l'altra in un unico messaggio! 13:25

Entrare auguri pagamento netto accademico 13:26

Dopo averci riflettuto parecchio, penso che la soluzione sia:

INTERESSE 🎉 13:26

(b)

Fig. 1. Screenshots of the implemented Telegram chatbot when (a) the correct sixth word is returned, and (b) a wrong sixth word is returned.

Fig. 1 shows two screenshots of the Telegram chatbot. Fig. 1a shows an instance of the game in which the artificial player found the correct sixth word. The five words were:

– *Amore* (Italian for *love*)
– *Fascia* (Italian for *sash*)
– *Oro* (Italian for *gold*)
– *Tenera* (Italian for *tender*)
– *Maggiore* (Italian for *major*)

The correct sixth word, as proposed by the artificial player, was *età* (Italian for *age*). As a matter of fact, one can say in Italian: *età dell'amore* (Italian for *age of love*); *fascia d'età* (Italian for *age range*); *età dell'oro* (Italian for *golden age*); *tenera età* (Italian for *early age*); and *maggiore età* (Italian for *age of majority*).

Fig. 1b shows an instance of the game in which the artificial player did not find the correct sixth word. The five words were:

– *Entrare* (Italian for *enter*)
– *Auguri* (Italian for *wishes*)
– *Netto* (Italian for *net*)
– *Pagamento* (Italian for *payment*)
– *Accademico* (Italian for *academic*)

In this case, the correct sixth word was *ritardo* (Italian for *delay*). As a matter of fact, one can say in Italian: *entrare in ritardo* (Italian for *enter late*); *auguri in ritardo* (Italian for *belated wishes*); *in netto ritardo* (Italian for *in net delay*); *pagamento in ritardo* (Italian for *late payment*); and *ritardo accademico* (Italian for *academic quarter*). Instead, the sixth word proposed by the artificial player was *interesse* (Italian for *interest*). Although the proposed sixth word was wrong, it is worth noting that it is strongly related to some of the five words. As a matter of fact, one can say in Italian: *interesse netto* (Italian for *net interest*) and *pagamento di interesse* (Italian for *interest payment*).

5 Conclusion

This paper discussed the design of an artificial player for a word game that has been used as closing game in a popular Italian television show since 2005. Given five words, the goal of the player is to find a new word related to, but also different from, the given five words within a short period of time (normally, one minute). The challenge to design and implement an artificial player for this game was proposed at EVALITA 2020 [1]. Only two other artificial players were developed to accept the challenge, namely *Il Mago della Ghigliottina* [12] and *GUiLlotine gLovE resolVER* (*GUL.LE.VER.*) [6]. Even if the artificial player discussed in this paper did not participate to EVALITA 2020, a comparison with the two players is deserved. The preliminary experimental results presented in the previous section show that the success rate of the artificial player discussed in this paper is 24%. The success rate of Il Mago della Ghigliottina is 68.6% [12], and the success rate of GUL.LE.VER. is 26% [6]. Note that the mentioned success rates were not obtained using a common set of game instances, and therefore their relevance to compare the artificial players is limited.

It is worth noting that human players often fail when they play the considered game. Hence, while a success rate of 24% seems low, it can be considered encouraging. Therefore, various developments have already been planned to improve the success rate of the discussed player. Future developments of the player include the extension of the dataset and the use of additional metrics to compare words. For example, the introduction of appropriate weighting factors is planned to give more relevance to the pairs of words obtained from compound words and idiomatic phrases. Finally, a detailed analysis of the game instances in which the player fails is planned for the near future to better understand the strengths and the weaknesses of the adopted approach.

References

1. Basile, P., Lovetere, M., Monti, J., Pascucci, A., Sangati, F., Siciliani, L.: Ghigliottin-AI @ EVALITA 2020: Evaluating artificial players for the language game “La Ghigliottina”. In: Basile, V., Croce, D., Di Maro, M., Passaro, L.C. (eds.) Seventh Evaluation Campaign of Natural Language Processing and Speech Tools for Italian. Final Workshop (EVALITA 2020). CEUR Workshop Proceedings, vol. 2765. RWTH Aachen (2020)
2. Belinkov, Y., Glass, J.: Analysis methods in neural language processing: A survey. Transactions of the Association for Computational Linguistics **7**, 49–72 (2019)
3. Bitter, C., Elizondo, D.A., Yang, Y.: Natural language processing: A Prolog perspective. Artificial Intelligence Review **33**(1), 151–173 (2010)
4. Chater, N., Manning, C.D.: Probabilistic models of language processing and acquisition. Trends in Cognitive Sciences **10**(7), 335–344 (2006)
5. Dahl, V.: Natural language processing and logic programming. The Journal of Logic Programming **19-20**, 681–714 (1994)
6. de Francesco, N.: GUL.LE.VER @ GhigliottinAI: A glove based artificial player to solve the language game “La Ghigliottina”. In: Basile, V., Croce, D., Di Maro, M., Passaro, L.C. (eds.) Seventh Evaluation Campaign of Natural Language Processing and Speech Tools for Italian. Final Workshop (EVALITA 2020). CEUR Workshop Proceedings, vol. 2765. RWTH Aachen (2020)
7. Lyding, V., Stemle, E., Borghetti, C., Brunello, M., Castagnoli, S., Dell’Orletta, F., Dittmann, H., Lenci, A., Pirrelli, V.: The PAISÀ corpus of Italian Web texts. In: Proceedings of the 9th Web as Corpus Workshop (WaC-9), pp. 36–43. Association for Computational Linguistics (2014)
8. Manning, C.D., Schutze, H.: Foundations of Statistical Natural Language Processing. MIT Press (1999)
9. Mooney, R.J.: Inductive logic programming for natural language processing. In: Muggleton, S. (ed.) Inductive Logic Programming. pp. 1–22. Springer (1997)
10. Riguzzi, F., Bellodi, E., Lamma, E., Zese, R., Cota, G.: Probabilistic logic programming on the Web. Software Practice and Experience **46**(10), 1381–1396 (2016)
11. Riguzzi, F., Lamma, E., Alberti, M., Bellodi, E., Zese, R., Cota, G.: Probabilistic logic programming for natural language processing. In: Chesani, F., Mello, P., Milano, M. (eds.) AI*IA Workshop on Deep Understanding and Reasoning: A Challenge for Next-generation Intelligent Agents. CEUR Workshop Proceedings, vol. 1802, pp. 30–37. RWTH Aachen (2016)
12. Sangati, F., Pascucci, A., Monti, J.: “Il Mago della Ghigliottina” @ GhigliottinAI: When linguistics meets artificial intelligence. In: Basile, V., Croce, D., Di Maro, M., Passaro, L.C. (eds.) Seventh Evaluation Campaign of Natural Language Processing and Speech Tools for Italian. Final Workshop (EVALITA 2020). CEUR Workshop Proceedings, vol. 2765. RWTH Aachen (2020)
13. Young, T., Hazarika, D., Poria, S., Cambria, E.: Recent trends in deep learning based natural language processing. IEEE Computational Intelligence Magazine **13**(3), 55–75 (2018)