MIGUS: An artificial player for the language game "La ghigliottina"

Semantics in intelligent systems exam

Francesco Benedetti

Prof. **Giovanni Semeraro**

Tutors: Prof. **Pierpaolo Basile**, Prof. **Marco De Gemmis**

a.a. 2021/2022

Abstract

La ghigliottina is the final challenge in the Italian TV quiz show *L'eredità*. Given 5 clues, the competitor must guess within one minute the word that semantically ties with all of them, in order to win the jackpot. Automatically solving this game and possibly outperforming humans is an interesting challenge for NLP systems. That is why it became one of the tasks for the 2018 and 2020 EVALITA editions, during the competition named "NLP4FUN". This work describes MIGUS (**M**utual **I**nformation **GU**illottine **S**olver), an attempt at this challenge.

1 Introduction

In the game *La ghigliottina*, the competitor starts from five clues, consisting of words that can be related to any topic, and the winning word has different types of relations with the five clues. The correlations never require a deep knowledge in a specific field in order to be understood. The semantic associations can consist in Italian ways of saying, movie names, proverbs, song names and so on. Hence competitors whose knowledge ranges through several cultural fields are advantaged with respect to those who know few fields, even if they know them very well.

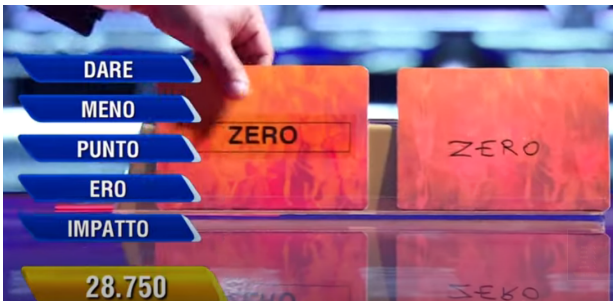

Figure 1: *Figure 1: An example of the game.*

Figure 1 depicts an example of game instance. The competitor won the jackpot by guessing the correct word. *Dare zero* (Give zero) means evaluating a very bad work with zero mark. *Valere meno di zero* (be worth less than zero) is something said to people who act without honor. *Punto zero* (zero point) is another way to refer to the origin of a Cartesian plane. *Ero zero* is the name of an Italian music disk by Renato Zero. *Impatto zero* (zero impact) refers to vehicles which don't pollute the environment. Af-

ter writing the guess, the players explain to the host and the people watching at home how they came up with their solution. During this phase, the host still tries to help them by giving additional clues (but the competitor cannot change the word he/she wrote. It is just to keep the suspense). Many times happens that the competitor wrote the wrong word but, during the "review process", guesses the correct one, which just didn't come up in mind in the time slot in which he/she had to think to a solution. The clues are words that relate to several fields, and for a human it is tricky to remember all of them in just one minute. This is why human performance in this game is, on average, of one victory every five attempts (20%). Computers don't have problems in remembering something, and their ability in efficiently dealing with tons of information is something that can be very helpful when trying to solve this game.
The system being proposed heavily relies on the concept of Multi-Word Expressions (MWE) and compounds expressions which, after an analysis of some game instances, turned out to be the most popular tie of association between the clues and the solution.

2 Related works

Four different systems have been proposed for solving *La ghigliottina*. Chronologically speaking, **OTTHO**[1][2] was the first attempt. It is based on the concept of knowledge infusion, which is defined in the paper as *"the process of providing a system with background knowledge that gives it a deeper understanding of the information it deals with"*. The authors try to reproduce in their system the concept of Cognitive Units, the way in which information is stored in the human's brain. Each CU is represented as a structured document containing words and expressions related to the same semantic concept. The CUs are created during the training phase following a set of heuristics. For each game run, a Spreading Activation Network (SAN) is used to retrieve the solution word. The nodes representing the five hints are added to the network, which is expanded with the CUs that are most similar to the hints. Each CU is connected to the hint by an oriented link, weighted with the cosine similarity of the word with the CU. Each node i is then associated with an activation value $A_i(p)$, initially set to 0 for all the nodes except the clues (whose activation value is set to 1) that, at each iteration p, is updated by a function. The iterations are two, and at the end of the process the nodes with the highest activation values are ranked in descending order and proposed as solutions.
The approach proposed by [3] (**Il mago della ghigliottina**) is based on the concept of MWE. A MWE is a combination of words which, when are together, assume a meaning that is different from their meaning if considered as single words. An example is the common way of saying *to kick the bucket* when referring to a person who died. The authors used a set of predefined patterns that two words must match in order to be detected as MWE. A co-occurrence matrix is then built to store how many times a pair of words is met in the corpus. The third step is to create another matrix that stores, for each pair of words w_i and w_j, their Pointwise Mutual Information (PMI). After being provided with the five clues, the system will look for the word that maximizes the score obtained by summing the PMI between each clue word and the candidate word. The procedure adopted by MIGUS is partly inspired from the one just depicted, with some differences.
Also **Robospierre** [4] borrows something from **Il mago della ghigliottina**, but using association rules instead of relying on the PMI.
Gullever [5] introduces an even different approach involving word embeddings. First, a Glove [6] model is learned in order to produce, for each word, an embedding representation. Then, for each of the five clues, the algorithm selects the embeddings of the closest words, using cosine similarity as measure. The obtained set of words is filtered to remove words that can't be solutions (like verbs and conjunctions) and ranked via an apposite function. The ranked list of solutions is then returned.
Lastly, the idea of **Luca Squadrone** [7] is knowledge base-driven. It uses *Morph-it*, an Italian morphological resource consisting of a lexicon of inflected forms with their lemma and morphological features. It is queried with the five hints to provide a list of candidate solutions. The list is then shrinked by looking

for proverbs, book/movie titles, ways of saying etc. that contain both the clues and, on turn, the different candidate solutions. The word with the most co-occurrences with the hints is then proposed as solution.

3 Proposed procedure

3.1 Corpora

As previously stated, the adopted procedure is built on the concept of MWE and compounds. Differently from MWEs, compounds are words that are put together to form a new expression, while maintaining their meaning. Examples are *pesce spada* (swordfish) or *traffico intenso* (intense traffic). From now on, when talking aboout MWEs we will refer to both MultiWord Expressions and compounds.
A very large corpus is required, in order to retrieve as many expressions as possible. The corpus that has been used was obtained by merging the following two corpora:

1. The **Paisà corpus** [8]: it is a collection of Italian documents taken from the web. It contains circa 250M tokens. It is available in four versions, and I used is the largest one[1] since it is in the CoNLL format: each word, is annotated with additional information, but I was interested only in the lemma and the POS tag.

2. A dump of Italian **Wikipedia articles**, downloaded via the Wikimedia website. The version that was used was dumped on 20/12/2021[2]. The WikiExtractor tool[3] was then used to clean the downloaded corpus and structure it in json format. A further processing phase was executed by performing lemmatization and POS tagging for each word in order to produce files with the same structure as the ones in the Paisà corpus.

The corpus is processed looking for MWEs. The assumptions that are made derive from the patterns that the authors of [3] look for and also from the patterns analyzed in [9], and are the following:

- A MWE can start with either a noun, adjective, verb;
- A MWE can end with either a noun, adjective, verb, negation, adverb, proper noun;
- A MWE cannot be made of two adjectives;
- Between the head and the tail of the MWE, a maximum of two words (like conjunctions or prepositions) can occur;
- No punctuation can occur in a MWE.

Moreover, all the words tagged as SW (meaning they do not belong to the Italian vocabulary), and those containing characters that the Italian language does not use (so anything that is not a letter of the alphabet or an accentuated letter) are ignored.
The entire corpus is scanned and a dictionary is built. It has as keys the MWE, in the format **word1_word2** and as values the number of occurrences of that MWE. For instance, if the sequence *Finire agli arresti domiciliari* (end up under house arrest) is met, the retrieved MWEs will be **finire_arresti** and **arresti_domiciliari**, and the values corresponding to those keys will be incremented. Of course, if the keys are not present in the dictionary, they will be added and their value will be set to 1. This dictionary is called **mwe_dict**.
Three more dictionaries are created. The first stores, for each single word, its number of occurrences in the corpus. The second, named **word2index_mapping**, associates each word to a progressive integer that will work as ID for that word. The third, named **index2word_mapping**, allows to retrieve a word by providing its ID.

3.2 Resolution strategy

The first step in the resolution strategy is to retrieve from **mwe_dict** only those keys containing the provided clue words. By doing so, we work with a portion of the original dictionary that does not contain those MWEs that would be useless since they

[1] https://clarin.eurac.edu/repository/xmlui/handle/20.500.12124/3
[2] http://itwiki-20220101-pages-articles-multistream.xml.bz2
[3] https://github.com/attardi/wikiextractor

couldn't help in the solving process. After this, for one hint, the following steps are performed:

1. Build an array whose dimension evens the dimension m of the set of words in the corpus. This array is initially filled with zeros.

2. At the position j, the array will contain the Mutual Information (MI) between the clue and the word whose ID is j. That word can be retrieved via the **index2word_mapping** dictionary. Let:

   (a) C_{cj} the number of co-occurrences between the clue, denoted as c, and the j^{th} word;

   (b) O_c the occurrences of the clue in the corpus;

   (c) O_j the occurrences of the j^{th} word in the corpus;

   (d) N the total number of words in the corpus; The **expected frequency** E_{cj} of the co-occurrence between the clue and the j^{th} word is given by the formula:

   $$E_{cj} = \frac{O_c * O_j}{N}$$

   And the **mutual information** MI between the clue and the j^{th} word is the ratio:

   $$MI = log_2(\frac{C_{cj}}{E})$$

   The MI was chosen because of its ability in providing a direct estimate of the association between two words.

3. Repeat from point 1 for each of the hints. At the end of the process, five arrays will have been produced, and they can be put together in a matrix M having five rows and m columns.

4. Sum the rows of M to obtain an array w.

5. An aspect that plays a very important role when evaluating a candidate solution is the number of hints with which it co-occurs. Indeed, the solution is that word that has a tie with all the five clues. By relying only on the MI, if there is a candidate solution that always co-occurs with only one clue, its MI will be very high, even if it never occurs with the other four clues. We need way to boost up the score for the solutions as the number of hints with which they co-occur increases. Therefore, an array v of dimension m is created. It is initialized as a zero array. Then, we loop through it. At the position k, we will assign the value 4 or 5 depending on the number of values greater than zero that are present in the k^{th} column in the matrix M, and that will represent the existence of a match between the word and the clue. By doing this, we are stating that, for a word to be a candidate solution, it has to occur with at least 4 hints.

6. The final score array s is obtained by performing a pointwise multiplication between w and v.

7. Sort in a descending order the values in s, but remembering their indexes in the original array.

8. Use the **index2word_mapping** dictionary to retrieve the words associated to the indexes of the first 10 values in the sorted array

9. Return the list of the words, which will be the proposed solutions ranked by their score.

4 Experiments

The proposed strategy was tested on a set of 288 game instances, collected in an XML file. The games belong to several editions of the TV show. In fact, over the years, the game's creators made it more challenging. As stated in the previous section, for each instance the system produces a ranked list of ten solutions. Of course it is preferable that the actual solution is put in the top positions of the list.
The **precision** is computed to evaluate the system. Suppose to mark a game as solved if the actual solution appears in the list. The number of solved games is SG. The precision is provided by the ratio:

$$\frac{|SG|}{|games|}$$

The **P@k** metric provides the precision by taking into account only the first k elements of the output lists. The **P@1** and **P@5** are calculated.
The Mean Reciprocal Rank (MRR) is used to score

the ability of the system in placing the solution at the top spots of the list:

$$MRR = \frac{\sum_{i=1}^{|games|} \frac{1}{r_i}}{|games|}$$

where r_i is the rank of the solution in the list for the i^{th} game instance.

5 Results

The system, on average, takes 45 seconds to produce a list of solutions for a game. This is below the 60 seconds time limit that humans are given to find a solution. Most of this time is spent in loading in memory the dictionaries. The evaluation metrics were performed for five different cases: the first considered all the MWEs in the dictionary. The second considered only the MWEs with at least five occurrences. The third, only MWEs with at least 10 occurrences. The fourth, MWEs with at least 15 occurrences and, lastly, MWEs with at least 30 occurrences. These trials were made to verify whether the solution process vould benefit the removal of rare MWEs. Another measure was calculated: the coverage of the words in the dictionary with respect to the clues provided. Indeed, there are some clue words that don't appear in the corpus or that were removed during the preprocessing phase. They mainly consist of digits, numbers and multiword clues. Table 1 contains the results (the **T** column indicates the occurrences threshold):

T	P@1	P@5	P@10	MRR	COV
0	0.313	0.409	0.451	0.36	98.1%
5	0.313	0.409	0.451	0.36	97.67%
10	0.313	0.409	0.451	0.36	97.46%
15	0.313	0.409	0.451	0.36	97.18%
30	0.313	0.409	0.451	0.36	96.47%

Table 1: *Comparison of performances after putting thresholds*

It is clear how putting constraints on the occurrences of a MWE does not have any impact on the performance. The only measure that is affected is the coverage. Some other modifications were tried to see if they could help increasing accuracy:

- Since numbers (intended as a word tagged as N, like "one", "two" and so on) can occur as hints, and in some cases they are the solution, I tried allowing them to the dictionary. However, accuracy stayed the same.
- A variant of the MI, named **local mutual information** was tried. It consists in multiplying the mutual information between two words for the number of occurrences of the MWE composed by the words. This worsened performance, which dropped to very low values, and hence the variant was discarded. Applying local MI gives a huge boost to those MWEs which are frequent in the corpus, leading the system to provide wrong solutions in many cases.

5.1 Comparison with OTTHO

In order to compare my system with OTTHO I tested it on the same games on which OTTHO was tested. It consists in subset of the one described in the Experiments section. It contains 266 games. The results obtained by the two systems are in reported in Table 2. On the dataset, MIGUS achieved **MRR = 0.372**

System	P@1	P@5	P@10
OTTHO	0.28	0.4	0.43
MIGUS	0.33	0.424	0.46

Table 2: *OTTHO vs MIGUS*

5.2 Test on the Evalita dataset

EVALITA is a periodic evaluation campaign of Natural Language Processing (NLP) and speech tools for the Italian language. In 2018 and 2020 the NLP4FUN competition took place during the campaign. It was aimed at automatically solving some language games, including *La ghigliottina*. **GULLEVER, Il mago della ghigliottina** and the system by **Luca Squadrone** joined the contest in its two editions, achieving the results reported in

Table 3.

System	P@1	MRR
Mago ghigliottina	0.686	0.643
GULLEVER	0.269	NP[4]
Squadrone	0.257	0.013

Table 3: *EVALITA participants performances*

The systems were evaluated on a sample of 350 game instances, for each of which they had to provide their best candidate solution. More details in [10] and [11]. MIGUS was tested on the same set. Table 4 contains the complete set of evaluation metrics achieved by the system.

P@1	P@5	P@10	MRR
0.369	0.546	0.603	0.442

Table 4: *Performance of MIGUS on the EVALITA test set*

6 Error analysis

In this section some of the errors made by the system are discussed:

6.1 Recurring solutions

There are some instances for which the system puts, in the list of candidate solutions, the same words, which are not even related to the clues. For instance:
Game id: 14
Clues: senza, camera, ragazza, quattro, roulette
Solution: pari

Game id: 17
Clues: quaranta, scalfaro, perdonare, porta, donna rosa
Solution: sette

A motivation for putting constraints on the occurrences of a MWE was to check if removing less popular MWEs could solve this problem.
Lists of solutions for the two games returned by the system without putting constraints on the number of occurrences of a MWE:
First game: vince, grazie, grande, crognolo, sipotrebbe, espasperanti, alievo, divantato, dilungherei, carpivo

Second game: crognolo, alievo, dilungherei, carpivo, concludessero, semicontemporanea, spioler, luzzi, sipotrebbe, mellifuo

It is noticable how words like *crognolo, carpivo, alievo, dilungherei* are present in both the lists, and they can be found (as well as some other words) in a bunch of lists returned for other game instances. Now, the solutions calculated by the system which used only MWEs appearing at least 5 times:
First game: vince, grazie, grande, senbatsu, solidissime, rinchiudiamo, coprirono, pensierosa, instacabile, imbracature
Second game: senbatsu, shiva, videocorso, risponderemo, congruenza, generativi, rinchiudiamo, coprirono, pensierosa, instacabile

Here the words mentioned before do not appear but are in some way replaced by *senbatsu, rinchiudiamo, coprirono, pensierosa, instacabile*
Maybe this happens because in both the instances took under analysis there is one clue that is not present in the dictionary, making the prediction strategy much weaker. However, there are also other cases where the system works smoothly even if one clue is missing.

6.2 Wrong POS

In some cases the system makes an error because it writes on top of the list a declined version of the correct answer, mistaking the part of speech. Here are some examples:
Clues: coperto, compagnia, auto, agente, vita
Solution: assicurazione
Predictions (first two spots): assicurazioni, assicurazione

Clues: fare, luce, porto, sostegno, tiro **Solution**: palo
Prediction (first three spots): pali, intermittente, palo

It is very likely that this is due to the words in the first position co-occurring more times with the clues than the actual solutions.

6.3 Correct reasoning, wrong solution

There are situations where the system performs the correct reasoning but comes up with the wrong solution:
Clues: traffico, borsa, giorno, conto, cerimonia
Solution: chiusura
Predictions (first two spots): apertura, chiusura

The words *apertura* (opening) and *chiusura* (closing) have opposite meaning and can be used interchangeably with all the five clues. This is a rather unlucky situation: if we gave some humans the five clues and asked each of them to choose the best solution between *apertura* and *chiusura*, we would probably end up having very discordant opinions. In this case, we can be satisfied with the system putting the two words in the first two places of the list.

Clues: telefono, velluto, giorno, cielo, topo
Solution: grigio
Predictions (first two spots): azzurro, grigio

Both *azzurro* (blue) and *grigio* (grey) are colors. *Telefono azzurro* (blue telephone) is an association for preserving children rights. This pattern actually makes more sense than *telefono grigio* (grey was the most popular color for old telephones). *Velluto* (velvet) can be either blue or grey and same applies for the sky (*cielo*) or a day (*giorno*). The association between *azzurro* and *topo* (mouse) does not make much sense since mouses are grey but, for the other clues, the reasoning is correct and makes sense.

7 Conclusions and future works

This report describes MIGUS, an artificial player for the game *La ghigliottina*. It exploits the concept of MultiWord Expression joint with the Mutual Information between two words. Differently from the other systems devoted to this task, it also gives much importance to the number of hints with which a potential solution co-occurs. The results are remarkable considering the approach is less complex than the others that were adopted and could be improved. MIGUS ranks second in accuracy, behind **Il mago della ghigliottina** which obtains astonishing results by following a similar approach. However it has the burden of having to compute a huge matrix, while MIGUS calculates a smaller matrix on the fly depending on the hints. The MRR value for MIGUS is remarkable and it is an indicator of the system's ability in placing the correct solutions on the top spots of the list.
To improve this system, some modifications could be made. The first one is enlarging the corpus. For memory limits of the available devices I couldn't use a larger corpus otherwise the MWE dictionary could have not been loaded. But using corpora containing proverbs, movie, song names and so on could help the system in improving the accuracy.
At the current state, too many MWEs are included in the dictionary because the selection criteria is very weak. By strengthening the criteria we would end up with a smaller vocabulary with less useless MWEs. An other improvement could consist in considering only the infinite form of the verbs. In fact, verbs are not very commonly given as clues, and almost never appear in a form different from the infinite. Hence, by considering this form we would reduce the size of the MWE dictionary and avoid considering useless words.

References

[1] BASILE, Pierpaolo, et al. OTTHO: on the tip of my thought. In: Joint European Conference on Machine Learning and Knowledge Discovery in Databases. Springer, Berlin, Heidelberg, 2009. p. 710-713.

[2] SEMERARO, Giovanni, et al. OTTHO: An Artificial Player for a Complex Language Game. In: PAI. 2012. p. 47-53.

[3] SANGATI, Federico; PASCUCCI, Antonio; MONTI, Johanna. Exploiting multiword expres-

sions to solve “la ghigliottina”. EVALITA Evaluation of NLP and Speech Tools for Italian, 2018, 12: 256.

[4] CIRILLO, Nicola; PERICOLO, Chiara; TUFANO, Pasquale. Robospierre, an Artificial Intelligence to Solve” La Ghigliottina”. In: CLiC-it. 2019.

[5] DE FRANCESCO, Nazareno. GUL.LE.VER @ GhigliottinAI: A Glove based Artificial Player to Solve the Language Game “La Ghigliottina”

[6] PENNINGTON, Jeffrey; SOCHER, Richard; MANNING, Christopher D. Glove: Global vectors for word representation. In: Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP). 2014. p. 1532-1543.

[7] SQUADRONE, Luca. Computer challenges guillotine: how an artificial player can solve a complex language TV game with web data analysis. EVALITA Evaluation of NLP and Speech Tools for Italian, 2018, 12: 262.

[8] LYDING, Verena, et al. The paisa'corpus of italian web texts. In: 9th Web as Corpus Workshop (WaC-9)@ EACL 2014. EACL (European chapter of the Association for Computational Linguistics), 2014. p. 36-43.

[9] SQUILLANTE, Luigi. Polirematiche e collocazioni dell'italiano. Uno studio linguistico e computazionale. Universitätsverlag Hildesheim, 2016.

[10] BASILE, Pierpaolo, et al. Overview of the evalita 2018 solving language games (nlp4fun) task. EVALITA Evaluation of NLP and Speech Tools for Italian, 2018, 12: 75.

[11] BASILE, Valerio, et al. Evalita 2020: Overview of the 7th evaluation campaign of natural language processing and speech tools for italian. In: 7th Evaluation Campaign of Natural Language Processing and Speech Tools for Italian. Final Workshop, EVALITA 2020. CEUR-ws, 2020.