A Retrieval Model for Automatic Resolution of Crossword Puzzles in Italian Language

Gianni Barlacchi[1] and **Massimo Nicosia**[2,1] and **Alessandro Moschitti**[2,1]
[1]Department of Information Engineering and Computer Science, University of Trento,
[2]Qatar Computing Research Institute
{gianni.barlacchi, m.nicosia, amoschitti}@gmail.com

Abstract

English. In this paper we study methods for improving the quality of automatic extraction of answer candidates for an extremely challenging task: the automatic resolution of crossword puzzles for Italian language. Many automatic crossword puzzle solvers are based on database system accessing previously resolved crossword puzzles. Our approach consists in querying the database (DB) with a search engine and converting its output into a probability score, which combines in a single scoring model, i.e., a logistic regression model, both the search engine score and statistical similarity features. This improved retrieval model greatly impacts the resolution accuracy of crossword puzzles.

Italiano. *In questo lavoro abbiamo studiato metodi per migliorare la qualità dell'estrazione automatica di risposte da utilizzare nella risoluzione di cruciverba in lingua italiana. Molti risolutori automatici utilizzano database di definizioni e risposte provenienti da cruciverba risolti in precedenza. Il nostro approccio consiste nell'applicare tecniche di Information Retrieval alla base di dati, accedendo a questa per mezzo di un motore di ricerca. Gli score associati ai risultati sono combinati con altre misure di similiarità in un singolo modello di regressione logistica, che li convertite in probabilità. Il risultante modello é in grado di individuare con piú affidabilità definizioni simili e migliora significativamente l'accuratezza nella risoluzione dei cruciverba.*

1 Introduction

Crossword Puzzles (CPs) are probably one of the most popular language game. Automatic CP solvers have been mainly targeted by the artificial intelligence (AI) community, who has mostly focused on AI techniques for filling the puzzle grid, given a set of answer candidates for each clue. The basic idea is to optimize the overall probability of correctly filling the entire grid by exploiting the likelihood of each candidate answer, fulfilling at the same time the grid constraints. After several failures in approaching the human expert performance, it has become clear that designing more accurate solvers would not have provided a winning system. In contrast, the Precision and Recall of the answer candidates are obviously a key factor: very high values for these performance measures would enable the solver to quickly find the correct solution.

Similarly to the Jeopardy! challenge case (Ferrucci et al., 2010), the solution relies on Question Answering (QA) research. However, although some CP clues are rather similar to standard questions, there are some specific differences: (i) clues can be in interrogative form or not, e.g., ≪Capitale d'Italia: *Roma*≫; (ii) they can contain riddles or be deliberately ambiguous and misleading (e.g., ≪Se fugge sono guai: *gas*≫); (iii) the exact length of the answer keyword is known in advance; and (vi) the confidence in the answers is an extremely important input for the CP solver.

There have been many attempts to build automatic CP solving systems. Their goal is to outperform human players in solving crosswords more accurately and in less time. Proverb (Littman et al., 2002) was the first system for the automatic resolution of CPs. It includes several modules for generating lists of candidate answers. These lists are merged and used to solve a Probabilistic-Constraint Satisfaction Problem. Proverb relies on a very large crossword database as well as several expert modules, each of them mainly based on domain-specific databases (e.g., movies, writers and geography). WebCrow (Ernandes et al.,

2005) is based on Proverb. In addition to its predecessor, WebCrow carries out basic linguistic analysis such as Part-Of-Speech tagging and lemmatization. It takes advantage of semantic relations contained in WordNet, dictionaries and gazetteers. Its Web module is constituted by a search engine, which can retrieve text snippets or documents related to the clue. WebCrow uses a WA* algorithm (Pohl, 1970) for Probabilistic-Constraint Satisfaction Problems, adapted for CP resolution. To the best of our knowledge, the state-of-the-art system for automatic CP solving is Dr. Fill (Ginsberg, 2011). It targets the crossword filling task with a Weighted-Constraint Satisfaction Problem. Constraint violations are weighted and can be tolerated. It heavily relies on huge databases of clues.

All of these systems queries the DB of previously solved CP clues using standard techniques, e.g., SQL Full-Text query. The DB is a very rich and important knowledge base. In order to improve the quality of the automatic extraction of answer candidate lists from DB, we provide for the Italian language a completely novel solution, by substituting the DB and the SQL function with a search engine for retrieving clues similar to the target one. In particular, we define a reranking function for the retrieved clues based on a logistic regression model (LRM), which combines the search engine score with other similarity features. To carry out our study, we created a clue similarity dataset for the Italian language. This dataset constitutes an interesting resource that we made available to the research community[1].

2 WebCrow Architecture

We compare our methods with one of the best systems for automatic CP resolution, WebCrow (Ernandes et al., 2005). It was kindly made available by the authors. The solving process is divided in two phases: in the first phase, the coordinator module forwards the clues of an input CP to a set of modules for the generation of several candidate answer lists. Each module returns a list of possible solutions for each clue. Such individual clue lists are then merged by a specific *Merger* component, which uses list confidence values and the probabilities of correctness of each candidate in the lists. Eventually, a single list of candidate-probability pairs is generated for each input clue. During the second phase WebCrow fills the crossword grid

[1] http://ikernels-portal.disi.unitn.it/projects/webcrow/

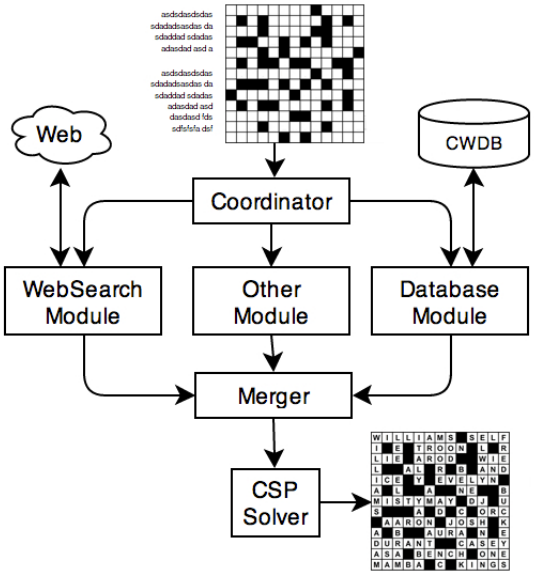

Figure 1: Overview of WebCrow's architecture.

by solving a constraint-satisfaction problem. WebCrow selects a single answer from each candidate merged list, trying to satisfy the imposed constraints. The goal of this phase is to find an admissible solution maximizing the number of correctly inserted words. In this paper, we focus on the DB module, and we describe it here.

Knowledge about previous CPs is essential for solving new ones. Indeed, clues often repeat in different CPs, thus the availability of a large DB of clue-answer pairs allows for easily finding the answers to previously used clues. To exploit the database of clue-answer pairs, WebCrow uses three different modules:

CWDB-EXACT, which simply checks for an exact match between the target clue and those in the DB. The score of the match is computed using the number of occurrences of the matched clue.

CWDB-PARTIAL, which employs MySQL's partial matching, query expansion and positional term distances to compute clue-similarity scores, along with the Full-Text search functions.

CWDB-DICTIO, which simply returns the full list of words of correct length, ranked by their number of occurrences in the initial list.

We compare our method with the CWDB-PARTIAL module. We improved it by applying a different retrieval function and using a linear model for scoring each possible answer.

3 Clue Retrieval from Database

This work is inspired by our earlier paper on learning to rank models for the automatic resolution of crossword puzzles for English language (Barlac-

Rank	Clue	Answer	Score
1	L'ente dei petroli	eni	8.835
2	Un colosso del petrolio	eni	8.835
3	Il petrolio americano	oil	8.835
4	Il petrolio della Mobil	oil	8.835
5	Il petrolio della Shell	oil	8.835

Table 1: Clue ranking for the query: *Il petrolio BP: oil*

chi et al., 2014). In that work, we showed that learning to rank models based on relational syntactic structures defined between the clues and the similar clue candidates can improve the retrieval of clues from a database of solved crossword puzzles. We cannot yet use our learning to rank model for the Italian language as we are implementing the needed syntactic/semantic parsers for such language. However, we have integrated the same search engine based on BM25 for Italian. Then, the completely new contribution is the use of supervised LRM to convert the Lucene scores into probabilities.

3.1 Clue Similarity for Italian language

WebCrow creates answer lists by retrieving clues from the DB of previously solved crosswords. As described before, it simply uses the classical SQL operator and full-text search. We verified the hypothesis that a search engine could achieve better results and we opted for indexing the DB of clues and their answers. We used the Open Source search engine Lucene (McCandless et al., 2010), its state-of-the-art BM25 retrieval model and the provided Italian Analyzer for processing the query. The analyzer performs basic operations, such as stemming and tokenization, over the input text. However, although this alone improved the quality of the retrieved clue list, a post-processing step is necessary for weighting the answer candidates appearing multiple times in the list. For example, Table 1 shows the first five clues, retrieved for a query originated by the clue: ≪Il petrolio BP: *oil*≫ (literally: *The petroleum BP*). Three answers out of five are correct, but they are not ranked before the others in the list. The Lucene scores of repeated candidates are not probabilities, thus their sum is typically not meaningful, i.e., it does not produce aggregated scores comparable between different answer candidates. For this reason, a LRM converts the Lucene score associated with each word into a probability. This way, we can sum the probabilities of the same answer candidates in the list and then normalize them considering the size of the list. We apply the following formula to obtain a single final score for each different answer candidate:

$$Score(G) = \sum_{c \in G} \frac{P^{LR}(y = 1|\vec{x}_c)}{n}$$

where c is the answer candidate, G is the set of answers matching exactly with c and n is the size of the answer candidate list. $\vec{x}_c$ is the feature vector associated with $c \in G$, $y \in \{0, 1\}$ is the binary class label ($y = 1$ when c is the correct answer). The conditional probability computed with the linear model is the following:

$$P^{LR}(y = 1|c) = \frac{1}{1 + e^{-y\vec{w}^T\vec{x}_c}}$$

where $\vec{w} \in \mathbb{R}^n$ is a weight vector (Yu et al., 2011).

In order to capture the distribution of the Lucene scores over the answer candidates list, we used the following simple features.

Lucene scores. These features are useful to characterize the distribution of the BM25 scores over the list. They include: the BM25 score of the target candidate and the maximum and minimum BM25 scores of the entire list. In particular, the last two features give the model information about the Lucene score range.

Rank. For each candidate answer c we include the rank $r \in [1, n]$ provided by the search engine Lucene. n is the size of the answer candidate list.

Clue distance. It quantifies how dissimilar the input clue and the retrieved clue are. This formula is mainly based on the well known Levenshtein distance.

For building the training set, we used a set of clues to query the search engine. We obtained candidates from the indexed clues and we marked them using the available ground truth. Clues sharing the same answer of the query clue are positive examples. During testing, clues are again used as search queries and the retrieved clue lists are classified.

4 Experiments

In this section, we present the results of our model. Our referring database of Italian clues is composed by 46,270 unique pairs of clue-answer, which belong to three different crossword editors.

4.1 Experimental Setup

For training the classifier we used Scikit-learn LRM implementation (Pedregosa et al., 2011) with default parameters. To measure the impact of the rerankers as well as the baselines, we used well known metrics for assessing the accuracy of

Model	MRR	AvgRec	REC@1	REC@5	REC@10
WebCrow	73.00	78.13	64.93	83.51	86.11
BM25	77.27	86.30	65.75	93.40	100.00
BM25+LRM	**81.20**	**88.94**	**71.12**	**95.70**	**100.00**

Table 2: Clue retrieval

QA and retrieval systems, i.e.: Recall at rank 1 (R@1, 5 and 10), Mean Reciprocal Rank (MRR), the average Recall (AvgRec). R@k is the percentage of questions with a correct answer ranked at the first position. MRR is computed as follows: $MRR = \frac{1}{|Q|}\sum_{q=1}^{|Q|}\frac{1}{rank(q)}$, where *rank(q)* is the position of the first correct answer in the candidate list. AvgRec and all the measures are evaluated on the first 10 retrieved clues.

4.2 Similar clue retrieval

We created the training and test sets using the clues contained in the database. The database of clues can be indexed for retrieving similar clues. The training set contains 10,000 clues whose answer may be found in the first ten position. With the same approach, we created a test set containing 1,000 clues that (i) are not in the training set and (ii) have at least an answer in the first ten position. We used the search engine to retrieve the clues in both training and test dataset creation.

We experimented with two simple different models: (i) BM25 and (ii) BM25 + LRM. However, since WebCrow includes a database module, in Tab. 2, we have an extra row indicating its accuracy evaluated using the CWDB-PARTIAL module. We note that in the BM25 model the list is ranked using the Lucene score while, in the BM25 + LRM the list is ranked using the probability score as described in the previous section. The result derived from the test set show that:
(i) BM25 is very accurate, i.e., an MRR of 77.27%. It improves on WebCrow about 4.5 absolute percent points, demonstrating the superiority of an IR approach over DB methods.
(ii) LRM achieves higher MRR, up to 4 absolute percent points of improvement over BM25 and thus about 8.5 points more than WebCrow.
(iii) Finally, the relative improvement on REC@1 is up to 9.5% (6.19% absolute). This high result is promising in the light of improving WebCrow for the end-to-end task of solving complete CPs.

4.3 Impact on WebCrow

In these experiments, we used our retrieval model for similar clues (BM25+LRM) using 5 complete CPs (for a total of 397 clues) created for a past Italian competition, organized by the authors of WebCrow. This way, we could measure the impact of our model on the complete task carried out by WebCrow. More specifically, we give our list of answers to WebCrow in place of the list that would have been extracted by the CWDB module . It should be noted that to evaluate the impact of our list, we disabled the WebCrow access to other lists, e.g., dictionaries. This means that the absolute resolution accuracy of WebCrow using our and its own lists can be higher (see (Ernandes et al., 2008) for more details).

Model	MRR	REC@1	REC@5	REC@10
WebCrow	30.89	27.63	35.17	36.14
Our Model	**34.41**	**29.36**	**36.92**	**38.93**

Table 3: Performance on the word list candidates averaged over the clues of 5 entire CPs

The first result that we derived is the accuracy of the answer list produced from the new data, i.e., constituted by the 5 entire CPs. The results are reported in Tab. 3. We note that the improvement of our model is lower than before as a non-negligible percentage of clues are not solved using the clue DB. Additionally, when we computed the accuracy in solving the complete CPs, we noted a small improvement: this happens because BM25 does not retrieve enough correct candidates for our specific test set constituted of five entire crossword puzzles.

5 Conclusions

In this paper, we improve the answer extraction from DB for automatic CP resolution. We combined the state-of-the-art BM25 retrieval model and an LRM by converting the BM25 score into a probability score for each answer candidate. For our study and to test our methods, we created a corpora for clue similarity containing clues in Italian. We improve on the lists generated by WebCrow by 8.5 absolute percent points in MRR. However, the end-to-end CP resolution test does not show a large improvement as the percentage of retrieved clues is not high enough.

Acknowledgments

We would like to thank Marco Gori and Marco Ernandes for making available WebCrow. The research described in this paper has been partially supported by the EU FP7 grant #288024: LIMOSINE – Linguistically Motivated Semantic aggregation engiNes. Many thanks to the anonymous reviewers for their valuable work.

References

Gianni Barlacchi, Massimo Nicosia, and Alessandro Moschitti. 2014. Learning to rank answer candidates for automatic resolution of crossword puzzles. *CoNLL-2014*.

Marco Ernandes, Giovanni Angelini, and Marco Gori. 2005. Webcrow: A web-based system for crossword solving. In *In Proc. of AAAI '05*, pages 1412–1417. Menlo Park, Calif., AAAI Press.

Marco Ernandes, Giovanni Angelini, and Marco Gori. 2008. A web-based agent challenges human experts on crosswords. *AI Magazine*, 29(1).

David A. Ferrucci, Eric W. Brown, Jennifer Chu-Carroll, James Fan, David Gondek, Aditya Kalyanpur, Adam Lally, J. William Murdock, Eric Nyberg, John M. Prager, Nico Schlaefer, and Christopher A. Welty. 2010. Building watson: An overview of the deepqa project. *AI Magazine*, 31(3):59–79.

Matthew L. Ginsberg. 2011. Dr.fill: Crosswords and an implemented solver for singly weighted csps. *J. Artif. Int. Res.*, 42(1):851–886, September.

Michael L. Littman, Greg A. Keim, and Noam Shazeer. 2002. A probabilistic approach to solving crossword puzzles. *Artificial Intelligence*, 134(1,Äì2):23 – 55.

Michael McCandless, Erik Hatcher, and Otis Gospodnetic. 2010. *Lucene in Action, Second Edition: Covers Apache Lucene 3.0*. Manning Publications Co., Greenwich, CT, USA.

F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. 2011. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830.

Ira Pohl. 1970. Heuristic search viewed as path finding in a graph. *Artificial Intelligence*, 1(3–4):193 – 204.

Hsiang-Fu Yu, Fang-Lan Huang, and Chih-Jen Lin. 2011. Dual coordinate descent methods for logistic regression and maximum entropy models. *Mach. Learn.*, 85(1-2):41–75, October.