Crossword Puzzle Resolution in Italian Using Distributional Models for Clue Similarity

Massimo Nicosia[†] and Alessandro Moschitti

[†]DISI, University of Trento, Povo (TN), Italy,
`massimo.nicosia@unitn.it`
Qatar Computing Research Institute, Doha, Qatar,
`amoschitti@qf.org.qa`

Abstract. Leveraging previous knowledge is essential for the automatic resolution of Crossword Puzzles (CPs). Clues from a new crossword may have appeared in the past, verbatim or paraphrased, and thus we can extract similar clues using information retrieval (IR) techniques. The output of a search engine implementing the retrieval model can be refined using learning to rank techniques: the goal is to move the clues that have the same answer of the query clue to the top of the result list. The accuracy of a crossword solver heavily depends on the quality of the latter. In previous work, the lists generated by an IR engine were reranked with a linear model by exploiting the multiple occurrences of an answer in such lists. In this paper, following our recent work on CP resolution for the English language, we create a labelled dataset for Italian, and propose (i) a set of reranking baselines and (ii) a neural reranking model based on distributed representations of clues and answers. Our neural model improves over our proposed baselines and the state of the art.

Keywords: distributional models, information retrieval, learning to rank

1 Introduction

Automatic solvers of CPs require accurate list of candidate answers for finding the correct solution to new clues. Candidate answers to a target clue can be found by retrieving clues from past games that are similar to the former. Indeed, the retrieved clues may have the same answers as the target clue. Databases (DBs) of previously solved CPs (CPDBs) are thus very useful, since clues are often reused or reformulated for building new CPs.

In this paper, we propose distributional models for reranking answer candidate lists generated by an IR engine. We present a set of baselines that exploit distributed representations of similar clues. Most importantly, (i) we build a dataset for clue retrieval for Italian, composed of 46,270 clues with their associated answers, and (ii) we evaluate an effective neural network model for computing the similarity between clues. The presented dataset is an interesting resource that we make available to the research community[1]. To assess the effec-

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

tiveness of our model, we compare it with the state-of-the-art reranking model for Italian [2].

The experimental results in this paper demonstrate that:

- distributed representations are effective in encoding reranking clue pairs and candidate answers;
- our neural network model is able to exploit the distributed representations more effectively than the other baseline models; and
- our models can improve over a strong retrieval baseline and the previous state-of-the-art system.

2 Clue Reranking for Solving CPs

In this section, we briefly present the ideas behind the CP resolution systems and the state-of-the-art models for reranking answer candidates.

2.1 CP Solvers

CP solvers are in many ways similar to question answering (QA) systems such as IBM Watson [5]. Indeed, their goal is not different: in order to find the correct answer for a given clue, candidate answers are generated and then scored according to more or less sophisticated strategies [9]. The main difference is the grid-filling step of CP solvers, which is casted as a Probabilistic Constraint Satisfaction Problem, e.g., [13]. In this step, the squares of the crossword puzzle are filled according to the crossword constraints. The possible combinations consider words from dictionaries or from the lists of answer candidates. Such lists can be generated by exploiting previously seen crossword puzzles or using subsystems specialized on domain-specific knowledge (e.g., famous persons, places, movies).

WebCrow is one of the best systems [4] for the resolution of CPs, and it relies on the aforementioned domain-specialized subsystems. In addition to that, it includes (i) a retrieval model for accessing clues stored in a database, (ii) a search module for finding answers from the Web, and (iii) a simple NLP pipeline.

Clearly, feeding the solver with high quality answer lists (i.e., lists containing the correct answers at the top) produces higher speed and accuracy in the grid-filling task. For this reason, a competitive CP solver needs accurate rankers.

2.2 Similar Clue Retrieval and Reranking

One important source of candidate answers is the DB of previously solved CPs. A target clue for which we seek the answer is used to query an index containing the clues of the DB. The list of candidate answers depends on the list of similar clues returned by the search engine (SE). The target clue, the candidate clues and their answers can be encoded into a machine learning model and the answers can be scored by rerankers. The goal of the latter is to understand which candidate clues are more similar to the target clue, and put the former at the top of the candidate list, assuming their similarity indicates they share the same answer.

The reranking step is important because often SEs do not retrieve the correct clues in the first results, i.e., the IR model is not able to capture the correct semantics of the clues and their answers.

In our work on Italian crosswords [2], we applied a logistic regression model to score aggregated sets of candidate clues with the same answer.

However, for English crosswords, we also (i) applied a pairwise reranking model on structural representations of clues [1]; (ii) designed a reranking model for aggregating the evidence coming from multiple occurrences of the same answers in a candidate list [12]; and (iii) combined a support vector machine reranking model with a deep neural network, which interestingly learns a similarity matrix M from the labelled data [16].

Motivated by the results in (iii), we applied the same distributional model for Italian. Unfortunately, the model was not effective due to the small number of available Italian clues. The same problem surfaces when using a small training set of English clues. To solve the issue, we opted not to learn the similarity matrix M from the data. Thus, we use a simpler neural network architecture and we feed similarity information directly into the model.

3 Distributional Models for Reranking Similar Clues

Previous methods for clue reranking use similarity features between clues based on lexical matching or other distances between words computed on the Wikipedia graph or the WordNet ontology.

Treating words as atomic units has evident limitations, since this approach ignores the context in which words appear. The idea that similar words tend to occur in similar contexts has a long history in computational linguistic [6]. In distributional semantics, a word is represented by a continuous vector, which is the result of counting some word statistics from a large corpus. In our work, we take advantage of modern methods for computing distributed representations of words and sentences, which may alleviate the semantic gap between clues, and therefore, induce better similarities.

The neural network model for measuring the similarity between clues is presented in Fig. 1, and it is essentially a Multilayer Perceptron (MLP), which is a simplification of our previous work [14, 16, 15]. Given the dimensionality d of the word embeddings, the main components are:

(i) sentence matrices $\mathbf{s}_{c_i} \in \mathbb{R}^{d \times |\mathbf{c}_i|}$ obtained by stacking the word vectors $\mathbf{w}_j \in \mathbb{R}^d$ of the corresponding words w_j from the input clues $\mathbf{c}_i$;
(ii) a distributional sentence model $f : \mathbb{R}^{d \times |\mathbf{c}_i|} \rightarrow \mathbb{R}^d$ that maps the sentence matrix of an input clue $\mathbf{c}_i$ to a fixed-size vector representations $\mathbf{x}_{c_i}$ of size d;
(iii) an input layer that is the concatenation of the fixed-size representation of the target clue $\mathbf{x}_{c_1}$, the similar clue $\mathbf{x}_{c_2}$, and a feature vector $\mathbf{fv}$;
(iv) a sequence of fully-connected hidden layers that capture the interactions between the distributed representations of the clues and the feature vector;
(v) a *softmax* layer that outputs probability scores reflecting how well the clues match with each other.

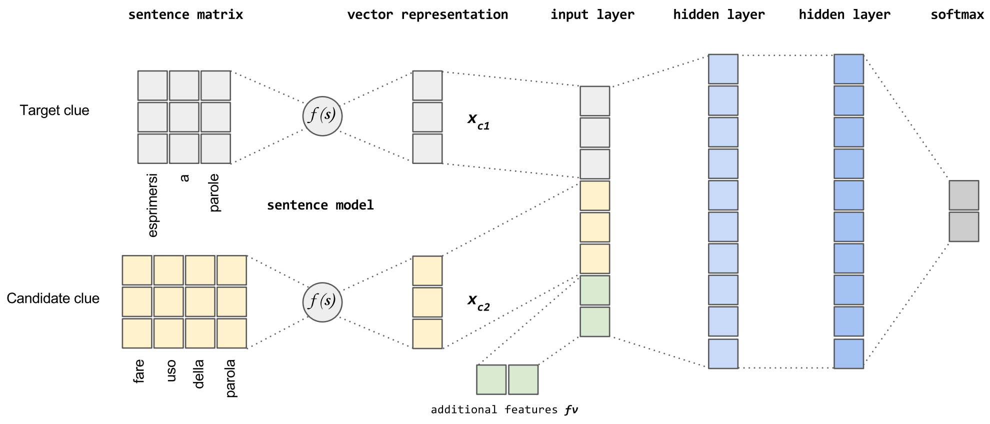

Fig. 1. Distributional sentence matching model for computing similarity between clues.

The choice of the sentence model plays a crucial role as the global representation of a clue contains the relevant information that the next layers in the network will use to compute the similarity between the clues.

Recently, distributional sentence models where $f(\mathbf{s})$ is represented by a sequence of convolutional-pooling feature maps, have shown state-of-the-art results on many NLP tasks, e.g., [7, 8].

Given the number of training instances in our dataset, we prefer to reduce the number of learning parameters. For this reason, we opt for a simple sentence model where $f(\mathbf{s}_{\mathbf{c}_i}) = \sum_i \mathbf{w}_i/|\mathbf{c}_i|$, i.e., the word vectors, are averaged to a single fixed-sized vector $\mathbf{x} \in \mathbb{R}^d$. In addition, our preliminary experiments for the English language revealed that this simpler model works just as well as more complicated single or multi-layer convolutional architectures.

4 Experiments

In this section we describe the experimental setting in which we evaluated our models. To conclude, we present the results of the experiments.

4.1 Experimental Setup

Data. The corpus of Italian crosswords contains 46,270 clues in the Italian language, with their associated answers, from *La Settimana Enigmistica* magazine, *La Repubblica* newspaper and the Web. In the original dataset, there are some clues containing words with hyphens in the middle. We normalize them by removing the hyphens whenever the cleaned word is in a list of terms extracted from the Italian Wiktionary[2]. In addition, we apply a simple tokenization rules to the clue definitions, in order to detach punctuation from the words. The processed dataset contains 46,185 unique clue/answer pairs. The unique definitions are 45,644, indicating that some definitions have multiple answer variations.

[2] https://it.wiktionary.org/

We construct our training and test datasets by sampling a set of training clues, and a disjoint set of test clues. We index only the training clues to prevent test clues from appearing in the training lists, and thus to avoid dependencies between training and test data. The indexing is performed with the Lucene library[3]. We enable the analyzer for Italian, which includes lowercasing, stemming and removal of stopwords. We query the SE with each training clue, obtaining related clues according to the BM25 retrieval model (we will refer to these clues as candidate lists). We of course remove the first exact match result (the training query clue is contained in the index), and retrieve only the clues whose answer length matches the answer length of the query clue. The candidate lists that do not contain the query answer in the first 10 positions are filtered out. Therefore, our lists always contain an answer, and have a maximum of 10 results. The test lists are constructed with the same process and constraints, by querying the training index with the test clues. The only difference is that we do not remove the first result, since the test clues are not present in the index.

Thus, our training and test instances consist of pairs of clues, i.e., a query clue and a similar candidate clue. The training set used in the experiments contains the results of 10,000 query clues, while the development and test sets contain the results of 1,000 query clues each. These numbers reflect the experimental setup of the previous state-of-the-art model for Italian.

Features. Our models use distributed representations of clue definitions and answers. Such representations are constructed from word embeddings [10]. The latter are learned by running the `word2vec` tool on the ItWaC corpus [3], a crawl of the Italian web containing 1,585,620,279 tokens. We use the SkipGram model trained with the hierarchical softmax algorithm. The dimensionality of the embeddings is set to 100, the window size to 5, and words with frequency less than 5 are filtered out.

The clue definitions are mapped to fixed-sized vectors by computing the average of their word embeddings, an approach also known as neural bag-of-words. It would be interesting to weight the word vectors by classical IR statistics associated with the corresponding words, but in this work, the vectors are unweighted.

In addition to the clue vectors, we use a set of features for capturing the SE result order, and the similarities between the distributed representations of clues and answers.

The *reversed rank* encodes the position of a candidate clue in the SE results. The rank is a decreasing value that starts at 10 for the top clue.

We also compute the *cosine similarity* of (i) the query and candidate clues, (ii) the query clue and the candidate answer, (iii) the candidate clue and the candidate answer. We obviously do not use the query answer since it is the gold label during testing. Therefore, the additional feature vector has 4 dimensions.

Distributional neural network model. Our neural network model classifies pairs of query and candidate clues as similar or not. The input to the model is a vector (of dimensionality 204) resulting from the concatenation of the distributed representations of the query and candidate clues, together with the

[3] https://lucene.apache.org/core/

Table 1. Results of the reranking experiments. The first section of the table contains the measures reported in previous state-of-the-art work. The second section of the table contains the evaluation measures of the models described in this paper.

Model	MRR	MAP	REC-1@1	REC-1@5
WebCrow [2]	73.00	-	64.93	83.51
BM25 [2]	77.27	-	65.75	93.40
LR on aggregated clues [2]	81.20	-	71.12	95.70
BM25	78.40	75.14	68.00	91.90
Cosine	79.46	75.92	69.10	93.10
LR classifier	84.09	81.60	75.10	96.40
DNN	**86.03**	**84.00**	**78.50**	**97.00**

feature vector. We use two hidden layers of size 256 and adopt the ReLU [11] as activation function. The model is regularized by applying dropout [17] on both hidden layers. Dropout prevents the co-adaptation of hidden units by setting to 0 a proportion p of the latter, during the forward pass at training time. In our case, we set p to 0.2.

The network is trained using Stochastic Gradient Descent (SGD) with shuffled mini-batches. The batch size is set to 16 examples. We train the model for 100 epochs with early stopping, i.e., stopping when the Mean Average Precision (MAP) on the development set does not increase for the last 7 epochs.

Evaluation. To measure the impact of the baseline models and our neural network model, we used well-known metrics for evaluating retrieval and QA systems: REC-1@k (@1, @5), Mean Average Precision (MAP) and Mean Reciprocal Rank (MRR). REC-1@k is the percentage of lists with a correct answer placed at the first position. Given a set of query clues Q, 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. MAP is the mean of the average precision scores for each query:

$$\frac{1}{Q} \sum_{q=1}^{Q} AveP(q).$$

4.2 Results

The first section of Table 1 contains the measures reported in [2], i.e., the previous state-of-the-art model. MAP values are not reported in the original work. The second section of the table contains the evaluation measures of our baselines and neural network model.

The **WebCrow** retrieval component establishes the matching between the query clue and the clues in its index using the standard SQL search operator. This explains its low performance compared to the other models.

Our **BM25** baseline is stronger due to the improved preprocessing of the definitions in the dataset of Italian clues.

The **Cosine** baseline scores each target and candidate clue pair by the cosine similarity of the distributed representations of the clues, i.e., the vectors obtained by applying $f(\mathbf{s}_{\mathbf{c}_i}) = \sum_i \mathbf{w}_i/|\mathbf{c}_i|$ to the sentence matrices of the two clues. Then, the pair in the lists are ordered by decreasing similarity.

Our **LR** baseline is a Logistic Regression classifier trained on the input layer, which is described in Section 3, together with the **DNN** neural network model.

The results show the effectiveness of the distributed representations of clues and their answers. Both supervised models benefit from this information, but the neural network, with its non-linearities, is able to better exploit the features fed to the models. With respect to the previous state-of-the-art model, the DNN produces 4.83% absolute and 5.95% relative improvement in MRR, and more interestingly, 7.38% absolute and 10.38% relative improvement in REC-1@1. This translates in more answers that are correctly selected and promoted to the top of the candidate answer list. Interestingly, the **Cosine** baseline is able to improve over the search engine.

The DNN model uses less features than the structural reranking models previously developed for this task, and does not require computationally expensive NLP for annotating the clues. As an interesting side note, we point out that the performance of the IR baseline for Italian are aligned with the performance of the IR baseline for English. This may suggest that, given enough training data, our simple model could perform even better, and we could be able to train a neural model with a learned similarity matrix M [16].

5 Conclusions

In this paper, we described the first distributional models for the retrieval of similar clues for crossword solving. We showed that distributed representations are effective for computing the similarity between clues, without involving expensive NLP and feature extractors in the reranking system. Our models outperform previous state-of-the-art system for the presented task, showing a consistent improvement across all the evaluation metrics.

We have described a dataset of clues and answers for Italian that we make available to the research community, together with our experimental models.

In the future, we plan to gather additional clue/answer pairs for Italian, in order to train more complex neural network models. Additionally, we will apply the models for question to question similarity in a question answering setting.

Acknowledgements

This work has been partially supported by the EC project CogNet, 671625 (H2020-ICT-2014-2, Research and Innovation action) and by an IBM Faculty Award. The first author is supported by the Google European Doctoral Fellowship 2015 in Statistical Natural Language Processing. Many thanks to the anonymous reviewers for their valuable suggestions.

References

1. Barlacchi, G., Nicosia, M., Moschitti, A.: Learning to rank answer candidates for automatic resolution of crossword puzzles. In: Proceedings of the Eighteenth Conference on Computational Natural Language Learning. Association for Computational Linguistics (June 2014)
2. Barlacchi, G., Nicosia, M., Moschitti, A.: A retrieval model for automatic resolution of crossword puzzles in italian language. In: First Italian Conference on Computational Linguistics (CLiC-it), Pisa, 09/12/2014-11/12/2014 (2014)
3. Baroni, M., Bernardini, S., Ferraresi, A., Zanchetta, E.: The wacky wide web: a collection of very large linguistically processed web-crawled corpora. Language resources and evaluation 43(3), 209–226 (2009)
4. Ernandes, M., Angelini, G., Gori, M.: Webcrow: A web-based system for crossword solving. In: In Proc. of the First Computational LinguisticsCLiC-it 2014. pp. 1412–1417. Menlo Park, Calif., AAAI Press (2005)
5. Ferrucci, D., Brown, E., Chu-Carroll, J., Fan, J., Gondek, D., Kalyanpur, A., Lally, A., Murdock, J.W., Nyberg, E., Prager, J., Schlaefer, N., Welty, C.: Building watson: An overview of the deepqa project. AI Magazine 31(3) (2010)
6. Firth, J.R.: A synopsis of linguistic theory 1930-55. 1952-59, 1–32 (1957)
7. Kalchbrenner, N., Grefenstette, E., Blunsom, P.: A convolutional neural network for modelling sentences. Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (June 2014)
8. Kim, Y.: Convolutional neural networks for sentence classification. In: Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP). pp. 1746–1751. Doha, Qatar (October 2014)
9. Littman, M.L., Keim, G.A., Shazeer, N.: A probabilistic approach to solving crossword puzzles. Artificial Intelligence 134, 23 – 55 (2002)
10. Mikolov, T., Sutskever, I., Chen, K., Corrado, G.S., Dean, J.: Distributed representations of words and phrases and their compositionality. In: Advances in Neural Information Processing Systems 26. pp. 3111–3119 (2013)
11. Nair, V., Hinton, G.E.: Rectified linear units improve restricted boltzmann machines. In: Proceedings of the 27th International Conference on Machine Learning (ICML-10). pp. 807–814 (2010)
12. Nicosia, M., Barlacchi, G., Moschitti, A.: Learning to rank aggregated answers for crossword puzzles. In: Advances in Information Retrieval - 37th European Conference on IR Research, ECIR, Vienna, Austria. Proceedings. pp. 556–561 (2015)
13. Pohl, I.: Heuristic search viewed as path finding in a graph. Artificial Intelligence 1(34), 193 – 204 (1970)
14. Severyn, A., Moschitti, A.: Learning to rank short text pairs with convolutional deep neural networks. In: Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval. pp. 373–382. SIGIR '15, ACM, New York, NY, USA (2015)
15. Severyn, A., Moschitti, A.: Modeling relational information in question-answer pairs with convolutional neural networks. In Preprint arXiv:1604.01178 (2016)
16. Severyn, A., Nicosia, M., Barlacchi, G., Moschitti, A.: Distributional neural networks for automatic resolution of crossword puzzles. In: Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (July 2015)
17. Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., Salakhutdinov, R.: Dropout: A simple way to prevent neural networks from overfitting. The Journal of Machine Learning Research 15(1), 1929–1958 (2014)