Get Your Model Puzzled: Introducing Crossword-Solving as a New NLP Benchmark

Anonymous ACL submission

Abstract

Solving crossword puzzles requires diverse reasoning capabilities, access to a vast amount of knowledge about language and the world, and the ability to satisfy the constraints imposed by the structure of the puzzle. In this work, we introduce solving crossword puzzles as a new natural language understanding task. We release a corpus of crossword puzzles collected from the New York Times daily crossword spanning 25 years and containing a total of 9152 puzzles, with an average of 85 clues per puzzle. These puzzles include a diverse set of clues: historic, factual, word meaning, synonyms/antonyms, fill-in-the-blank, abbreviations, prefixes/suffixes, wordplay, and cross-lingual, as well as clues that depend on the answers to other clues. We separately release the clue-answer pairs from these puzzles as an open-domain question answering dataset containing over half a million unique clue-answer pairs. For the question answering task, our baselines include several sequence-to-sequence and retrieval-based generative models. We also introduce a non-parametric constraint satisfaction baseline for solving the entire crossword puzzle. Finally, we propose an evaluation framework which consists of several complementary performance metrics.

1 Introduction

Recent breakthroughs in NLP established high standards for the performance of machine learning methods across a variety of tasks. However, even state-of-the-art models demonstrate fragility (Wallace et al., 2019) and exhibit sensitivity to shallow data patterns (McCoy et al., 2019; Zellers et al., 2019; Jin et al., 2020; Si et al., 2019; Sugawara et al., 2020; Yogatama et al., 2019; Niven and Kao, 2019). This has led to a growing demand for successively more challenging tasks.

One of the important tasks in natural language understanding is question answering (QA), with many recent datasets created to address different different aspects of this task (Yang et al., 2018; Rajpurkar et al., 2016; Kwiatkowski et al., 2019a; Zellers et al., 2019; Dua et al., 2019; Rogers et al., 2021). There are two forms of question answering (QA): extractive QA and open-domain QA. In extractive QA, a passage that answers the question is provided as input to the system along with the question. In open-domain QA, only the question is provided as input, and the answer must be generated either through memorized knowledge or via some form of explicit information retrieval over a large text collection which may contain answers.

The task of answering clues in a crossword is a form of open-domain question answering. Once a human or an open-domain QA system generates a few possible answer candidates for each clue, one of these candidates may form the correct answer to a word slot in the crossword grid, if the candidate meets the constraints of the crossword grid.

Solving a crossword puzzle is therefore a challenging task which requires (1) finding answers to a variety of clues that require extensive language and world knowledge, and (2) the ability to produce answer strings that meet the constraints of the crossword grid, including length of word slots and character overlap with other answers in the puzzle.

Our contributions in this work are as follows:

- We introduce a new natural language understanding task of solving crossword puzzles, along with a dataset of New York Times crosswords from Dec. 1, 1993 to Dec. 31, 2018.
- We propose an evaluation framework which consists of several complementary performance metrics.
- We release the collection of clue-answer pairs as a new open-domain QA dataset.
- We provide baselines for the proposed crossword task and the new QA task, including several sequence-to-sequence and retrieval-augmented generative Transformer models,

with a constraint satisfaction crossword solver.

2 Related Work

Our work is in line with open-domain QA benchmarks. Examples of such tasks include datasets where each question can be answered using information contained in a relevant Wikipedia article (Yang et al., 2015; Kwiatkowski et al., 2019a; Yang et al., 2018). Several QA tasks have been designed to require multi-hop reasoning over structured knowledge bases (Berant et al., 2013; Bordes et al., 2015). The main limitation of such datasets is that their question types are mostly factual. Crossword clues differ from these efforts in that they combine a variety of different reasoning types.

Another line of research that is relevant to our work explores the problem of solving Sudoku puzzles since it is also a constraint satisfaction problem. Most sudoku puzzles can be efficiently solved by algorithms that take advantage of the fixed input size and do not rely on machine learning methods (Simonis, 2005). The machine learning attempts for solving Sudoku puzzles have been inspired by convolutional (Mehta, 2021) and recurrent relational networks (Palm et al., 2017). Unlike Sudoku, however, crossword puzzles have arbitrary shape and internal structure and rely on answers to natural language questions that require reasoning over different kinds of world knowledge.

Solving crossword puzzles automatically has previously been studied by Ginsberg (2011); Littman et al. (2002); Keim et al. (1999) as constraint satisfaction problems(CSP). The Dr. Fill system proposed by Ginsberg (2011) treats each crossword puzzle as a singly-weighted CSP. However, Dr. Fill relied on a large set of historical clue-answer pairs (up to 5M) collected over multiple years from the past puzzles, using direct lookup, and a variety of heuristics. Similarly Littman et al. (2002) also use a variety of information retrieval modules to generate candidate answers. They find very poor crossword solving performance of their proposed weighted probabilistic CSP on ablations where they limit their answer candidate generator to not depend in any way on past clue-answer databases. Our goal in this work is to motivate solver systems to generate answers organically just like a human might, either from memory, using their world knowledge and language understanding, or by searching encyclopedic sources such Wikipedia or a dictionary.

3 Task and Dataset

For the purposes of our task, crosswords are defined as word puzzles with a given rectangular grid of white- and black-shaded squares. The goal is to fill the white squares with letters, forming words or phrases by solving textual clues which lead to the answers. The answer words and phrases are placed in the grid from left to right ("Across") and from top to bottom ("Down"). The shaded squares are used to separate the words or phrases. Usually, the white spaces and punctuation are removed from the answer phrases. A sample crossword puzzle is given in Figure 1. Note that the answers can include named entities and abbreviations, and at times require the exact grammatical form, such as the correct verb tense or the plural noun.

We divide the task of solving a crossword puzzle into two subtasks. The first subtask can be viewed as a question answering task, where a system is trained to generate a set of candidate answers for a given clue without taking into account any interdependencies between answers. The second subtask involves solving the entire crossword puzzle, i.e., filling out the crossword grid with a subset of candidate answers generated in the previous step.

The two tasks could be solved separately or in an end-to-end fashion. In the current work, we propose a separate solver for each task. We provide details on the challenges of implementing an end-to-end solver in the discussion section.

3.1 NYT Crossword Collection

Our dataset is sourced from the New York Times, which has been featuring a daily crossword puzzle since 1942. We worked with daily puzzles in the date range from December 1, 1993 through December 31, 2018 inclusive. All the crossword puzzles in our corpus are also available through the New York Times games website.[1]. We release two separate specifications of the dataset corresponding to the subtasks described above: the NYT Crossword Puzzle dataset and the NYT Clue-Answer dataset.

There are a few details that are specific to the NYT daily crossword. First, the clue and the answer must agree in tense, part of speech, and even language, so that the clue and answer could easily be substituted for each other in a sentence. Second, abbreviated clues indicate abbreviated answers. Further, clues that end in a question mark indicate a play on words in the clue or the answer. There are

[1] https://www.nytimes.com/crosswords

Figure 1: Crossword puzzle example. Highlighted clues with the corresponding answers match the clue types as described in Section 3.2 and are color coded in the same way like Figure 2. Source: NY Times daily crossword appeared on the July 7, 2009.

.

also a lot of short words that appear in crosswords much more often than in real life. These 3- and 4-letter words, referred to as crosswordese, can be very helpful in solving the puzzles. Finally, every Sunday through Thursday NYT crossword puzzle has a theme, something that unites the puzzle's longest answers. Theme answers are always found in symmetrical places in the grid.

Crossword Puzzle Dataset. The dataset consists of 9152 puzzles, split into the training, validation, and test subsets in the 80/10/10 ratio which give us 7293/922/941 puzzles in each set. We removed the total of 16/17 special puzzles from the validation and test splits respectively because they contained answers with multiple characters placed in the same cell (called rebus entries). We also removed 34/44 puzzles in the validation and test splits respectively since these puzzles have improvised rules for filling in answers into the grid for instance L shaped word slots instead of vertical and horizontal.

Most NYT crossword grids have a square shape of 15×15 cells, with the exception of Sunday-released crosswords being 21×21 cells. Other shapes combined account for less than 3% of the data. The vast majority of both clues and answers are short, with over 76% of clues consisting of a single word. For traditional sequence-to-sequence modeling such conciseness imposes an additional challenge, as there is very little context provided to the model. In most puzzles, over 80% of the grid cells are filled and *every* character is an intersection of two answers. Such high answer interdependency suggests a high cost of answer misprediction, as errors affect a larger number of intersecting words. More detailed statistics on the dataset are given in Table 1.

Clue-Answer Dataset. We generate an open-domain question answering dataset consisting solely of clue-answer pairs from the respective splits of the Crossword Puzzle dataset described above not removing the special puzzles. Within each of the splits, we only keep unique clue-answer pairs and remove all duplicates. However, certain clues may still be shared between the puzzles contained in different splits. We therefore remove from the training data the clue-answer pairs which are found in the test or validation data. This ensures that the model can not trivially recall the answers to the overlapping clues while predicting for the test and validation splits.

This produces the total of 578,275 clue-answer pairs, with 433k/72k/72k examples in the train/validation/test splits, respectively. Since certain answers consist of phrases and multiple words that are merged into a single string (such as "VERYFAST"), we further postprocess the answers by splitting the strings into individual words using a dictionary. Out of all the possible word splits of a given string we pick the one that has the smallest number of words. If there are multiple solutions, we select the split with the highest average word frequency.

3.2 Clue types

To provide more insight into the diversity of the clue types and the complexity of the task, we cate-

gorize all the clues into multiple classes, which we describe below.

Factual. Clues that encode encyclopedic knowledge and typically can be answered using resources such as Wikipedia (e.g. *South Carolina State tree: PALMETTO*). This type of clue is the closest to the questions found in open-domain QA datasets. Note that the facts required to solve some of the clues implicitly depend on the date when a given crossword was released. For instance, the clue "*President of Brazil*" has a time-dependent answer.

Historical. Clues that require the knowledge of historical facts and temporal relations between events. (e.g. *Automobile pioneer: BENZ*).

Word meaning. Clues that exploit general vocabulary knowledge and can typically be resolved using a dictionary. (e.g. *Opposing sides: FOES*).

Synonyms/Antonyms. Clues that focus on paraphrasing and synonymy relations (e.g. *Prognosticators: SEERS*). In most cases, such clues can be solved with a thesaurus.

Fill in the blank. Clues formulated as a cloze task (e.g. *Magna Cum __ : LAUDE*). Fill-in-the-blank clues are expected to be easy to solve for the models trained with the masked language modeling objective (Devlin et al., 2019).

Abbreviations. Clues answered with acronyms (e.g. *(Abbr.) Old Communist state: USSR*). Abbreviation clues are marked with "*Abbr.*" label.

Prefix/Suffix. Clues that suggest the answer is a suffix or prefix. (e.g. *Suffix with mountain : EER*)

Wordplay. Clues that rely on wordplay, anagrams, or puns / pronunciation similarities (e.g. *Consider an imaginary animal: BEAR IN MIND*). In a lot of cases, wordplay clues involve jokes and exploit different possible meanings and contexts for the same word.

Cross-lingual. Clues that either explicitly use words from other languages, or imply a specific language-dependent form of the answer. (e.g. *Sunrise dirección: ESTE*).

Clues dependent on other clues. Clues the answer to which can be provided only after a different clue has been solved (e.g. *Last words of 45 Across*). Although rare, this category of clues suggests that the entire puzzle has to be solved in certain order.

To understand the distribution of these classes, we randomly selected 1000 examples from the test split of the data and manually annotated them. Figure 2 illustrates the class distribution of the annotated examples, showing that the Factual class covers a little over a third of all examples. The synonyms/antonyms, word meaning and wordplay classes taken together comprise 50% of the data. The remaining 20% are taken by fill-in-the-blank and historical clues, as well as the low-frequency classes (comprising less than or around 1%), which include abbreviation, dependent, prefix/suffix and cross-lingual clues. We illustrate each one of these classes in the Figure 1.

	Train	**Validation**	**Test**
	Clue-Answer dataset		
# clues	4,33,033	72,303	72,939
avg/median clue length (words)	4.0/3	4.2/4	4.2/4
avg/median ans. length (chars)	5.5/5	5.7/5	5.6/5
avg/median ans. length (words)	1.3/1	1.3/1	1.3/1
	Crossword Puzzle dataset		
# puzzles	7,293	872	879
avg/median # of clues	83.5/76	83.6/76	82.9/76
avg cols×rows	15.9×15.9	15.9×15.9	15.8×15.8
% of cells filled	82.20%	80.20%	81.20%

Table 1: The full statistics on the two versions of the released datasets.

3.3 Evaluation metrics

In this section, we describe the performance metrics we introduce for the two subtasks.

Clue-Answer Task. For the clue-answer task, we use the following metrics:

- **Exact Match (EM)**. Model output matches the ground-truth answer exactly.
- **Contains (In)**. Model output contains the ground-truth answer as a contiguous substring

Since the ground-truth answers do not contain diacritics, accents, punctuation and whitespace characters, we also consider normalized versions of the above metrics, in which these are stripped from the model output prior to computing the metric. We will refer to them as $\mathbf{EM_{norm}}$ and $\mathbf{In_{norm}}$.

We report these metrics for top-k predictions, where k varies from 1 to 20.

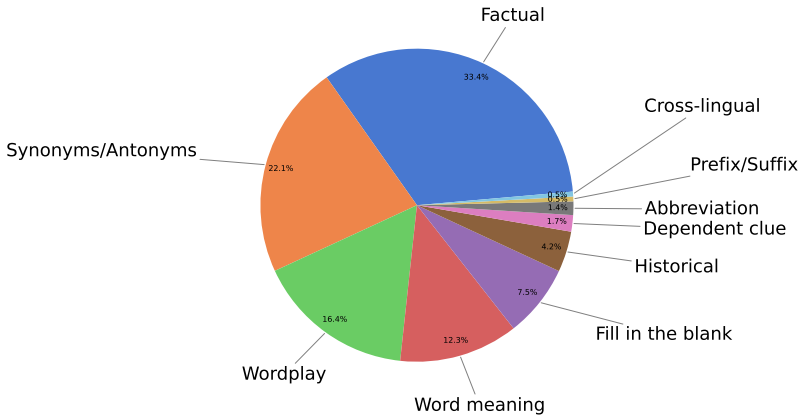

Figure 2: Class distribution of the 1000 manually annotated test examples.

Crossword Puzzle Task. To evaluate the performance of the crossword puzzle solver, we propose to compute the following two metrics:

- **Character Accuracy (Acc_{char}).** Percentage of characters in the predicted crossword solution that match the ground-truth solution.
- **Word Accuracy (Acc_{word}).** Percentage of words in the predicted crossword solution that match the ground-truth solution.

Since the question answering system might not be able to predict the right answers for some of the clues, it may only be possible to produce a partial solution to a puzzle. We propose two additional metrics to track what percentage the puzzle needs to be redacted to obtain a partial solution:

- **Word Removal (Rem_{word}).** % of words that need to be removed from the puzzle to produce a partial solution.
- **Character Removal (Rem_{word}).** % of characters that need to be removed from the puzzle grid to produce a partial solution.

4 Baselines

Our baseline approach is a two-step solution that treats each subtask separately. We first develop a set of baseline systems that solve the question answering problem, ignoring the grid-imposed answer interdependencies. We use seq-to-seq and retrieval-augmented Transformer baselines for this subtask. We feed generated answer candidates to a crossword solver in order to complete the puzzle and evaluate the produced puzzle solutions.

4.1 Clue-Answer Task Baselines

Sequence-to-sequence baselines. We fine-tune two sequence-to-sequence models on the clue-answer training data. We select two widely known models, BART (Lewis et al., 2019) and T5 (Raffel et al., 2019), which achieved state-of-the-art results on a set of generative tasks, including specifically abstractive QA involving commonsense and multi-hop reasoning (Fan et al., 2019; Khashabi et al., 2018; Zhang et al., 2018).

We train both models for 8 epochs with the learning rate of 5×10^{-5}, and a batch size of 60. [2]

Retrieval-augmented generation. T5 and BART store world knowledge implicitly in their parameters and are known to hallucinate facts (Maynez et al., 2020). Recently, a new method called retrieval-augmented generation (RAG) (Lewis et al., 2020) has been introduced for open-domain question answering. This method involves a Transformer encoder to encode the question and a decoder to generate the answer (Vaswani et al., 2017), but the encoded query is supplemented with relevant excerpts retrieved from an external textual corpus via Maximum Inner Product Search (MIPS); the entire neural network is trained end-to-end. Due to a built-in retrieval mechanism for performing a soft search over a large collection of external documents, such systems are capable of producing stronger results on knowledge-intensive open-domain question answering tasks than the vanilla sequence-to-sequence generative models

[2] We use BART-large with approximately 406M parameters and T5-base model with approximately 220M parameters, respectively.

	Top-1				Top-10				Top-20			
	EM	**EM$_{norm}$**	**In**	**In$_{norm}$**	**EM**	**EM$_{norm}$**	**In**	**In$_{norm}$**	**EM**	**EM$_{norm}$**	**In**	**In$_{norm}$**
T5-base	8.4	9.5	8.7	9.9	18.7	20.8	19.8	22.0	22.2	24.6	23.8	26.3
BART-large	13.8	16.1	15.0	17.6	31.0	36.7	32.4	38.0	34.0	40.1	35.3	41.3
RAG wiki	24.2	26.0	24.9	26.7	46.8	49.8	48.6	51.6	50.6	53.9	53.4	56.7
RAG dict	24.0	25.8	24.6	26.5	46.0	48.9	48.0	50.9	50.0	53.2	53.0	56.2

Table 2: Performance of baseline systems on the Clue Answering dataset. *EM* and *In* stand for the "Exact-match" and "Contains" metrics as described in Section 3.3. The computed metrics are shown for top-1, top-10, and top-20 predictions for a given model.

and are more factually accurate (Shuster et al., 2021). Motivated by this, we train RAG models to extract knowledge from two separate external sources of knowledge:

(a) **RAG-wiki** uses a full Wikipedia dump from December 2018. Following existing work Lewis et al. (2020); Karpukhin et al. (2020); Lee et al. (2019), each Wikipedia article is split into disjoint 100-word chunks, resulting in a total of 21M passages.

(b) **RAG-dict** uses several English dictionaries and thesauri sources, including Wiktionary[3], Merriam-Webster[4], and Google's English dictionary by Oxford Languages.[5]

For both of these models, we use the retriever embeddings pretrained on the Natural Questions corpus Kwiatkowski et al. (2019b) in order to prime the MIPS retrieval to return meaningful entries (Lewis et al., 2020). We train with a batch size of 8, label smoothing set to 0.1, dropout probability of 0.1, weight decay rate of 0.001, and a learning rate of 3×10^{-5} for 8 epochs.

4.2 Crossword Puzzle Task

A crossword puzzle can be cast as an instance of a satisfiability problem, and its solution represents a particular character assignment so that all the constraints of the puzzle are met. Under such formulation, three main conditions have to be satisfied: (1) the answer candidates for every clue must come from a set of words that answer the question, (2) they must have the exact length specified by the corresponding grid entry, and (3) for every pair of words that intersect in the puzzle grid, acceptable word assignments must have the same character at the intersection offset.

This class of problems can be modelled through Satisfiability Modulo Theories (SMT). SMT is a generalization of Boolean Satisfiability problem (SAT) in which some of the binary variables are replaced by first-order logic predicates over a set of non-binary variables. In the case of crosswords, a variable represents one character in the crossword grid which can be assigned a single letter of the English alphabet and 0 through 9 digit values. This is further subject to the constraints mentioned above which can be formulated with the equality operator and Boolean logical operators: `AND` and `OR`. For example, a word slot of length 3 where the candidate answers are "ESC", "DEL" or "CMD" can be formalised as:

$$\{v_1 = E \text{ AND } v_2 = S \text{ AND } v_3 = C\}$$
$$OR$$
$$\{v_1 = D \text{ AND } v_2 = E \text{ AND } v_3 = L\}$$
$$OR$$
$$\{v_1 = C \text{ AND } v_2 = M \text{ AND } v_3 = D\}$$

To solve the entire crossword puzzle, we use the formulation that treats this as an SMT problem. We use an open source implementation[6] of this formulation based on Z3 SMT solver de Moura and Bjørner (2008). The answer length and intersection constraints are imposed on the variable assignment, as specified by the input crossword grid.

We take the top-k predictions from our baseline models and for each prediction, select all possible substrings of required length as answer candidates. For simplicity, we exclude from our consideration all the crosswords with a single cell containing more than one English letter in it.

Our current baseline constraint satisfaction solver is limited in that it simply returns "not-satisfied" (`nosat`) for a puzzle where no valid solution exists, that is, when *all* the hard constraints of the puzzle are not met by the inputs. Since the candidate lists for certain clues might not meet all

[3] https://www.wiktionary.org/
[4] https://dictionaryapi.com/
[5] Accessed via https://dictionaryapi.dev/.
[6] https://github.com/pncnmnp/Crossword-Solver

the constraints, this results in a `nosat` solution for almost all crossword puzzles, and we are not able to extract partial solutions. To bypass this issue and produce partial solutions, we pre-filter each clue with an oracle that only allows those clues into the SMT solver for which the actual answer is available as one of the candidates.

5 Results

5.1 Clue-Answer Task

In Table 2 we report the Top-1, Top-10 and Top-20 match accuracies for the four evaluation metrics defined in Section 3.3.

Our results suggest high difficulty of the clue-answer dataset, with the best achieved accuracy metric staying under 30% for the top-1 model prediction. Even top-20 predictions have an almost 40% chance of not containing the ground-truth answer anywhere within the generated strings. Generative Transformer models such as T5-base and BART-large perform poorly on the clue-answer task, however, the model accuracy across most metrics almost doubles between T5-base (with 220M parameters) to BART-large (with 400M parameter).

Our strongest baseline, RAG-wiki and RAG-dict, achieve 50.6 and 50.0 exact-match accuracies on the clue-answer dataset, respectively. The In_{norm} score, which looks at whether any substrings in the generated answer match the ground truth – and which can be seen an upper bound on the model's ability to solve the puzzle – is slightly higher, at 56.7 for RAG-wiki and 56.2 for RAG-dict.

Not surprisingly, these results show that the additional step of retrieving Wikipedia or dictionary entries increases the accuracy considerably compared to the fine-tuned sequence-to-sequence models such as BART which store this information in its parameters. The normalized metrics which remove diacritics, punctuation and whitespace bring the accuracy up by 2-6%, depending on the model.

We examined the top-20 exact-match predictions generated by RAG-wiki and RAG-dict and find that both models are in agreement in terms of answer matches for around 85% of the test set. In other words largely both either correctly predict the ground truth or both fail to do so.

5.2 Crossword Puzzle Task

The baseline performance on the entire crossword puzzle dataset shows there is significant room for improvement of the existing architectures (see Table 3). Our best model, RAG-wiki, correctly fills in the answers for only 26% (on average) of the total number of puzzle clues, despite having a much higher performance on the clue-answer task, i.e. measured independently from the crossword grid (Table 2). This is explained by the fact that the clues with no ground-truth answer present among the candidates have to be removed from the puzzles in order for the solver to converge, which in turn relaxes the interdependency constraints too much, so that a filled answer may be selected from the set of candidates almost at random. Despite that, the baseline solver is able to solve over a quarter of each the puzzle on average.

Model	Solving Accuracy		Puzzle Removed	
	Acc$_{word}$	**Acc$_{char}$**	**Rem$_{word}$**	**Rem$_{char}$**
BART	16.6	28.4	55.6	43.4
RAG wiki	23.8	37.8	40.3	26.3
RAG dict	22.1	35.9	40.8	26.8

Table 3: Performance of baseline systems on the Crossword Puzzle dataset. We report the exact-match metric for top-20 predictions of the baseline models listed.

6 Qualitative analysis

Evaluation on the annotated subset of the data reveals that some clue types present significantly higher levels of difficulty than others (see Table 4). In particular, all of our baseline systems struggle with the clues requiring reasoning in the context of historical knowledge. As expected, all of the models demonstrate much stronger performance on the factual and word-meaning clue types, since the relevant answer candidates are likely to be found in the Wikipedia data used for pre-training. We observe the biggest differences between BART and RAG performance for the "abbreviation" and the "prefix-suffix" categories. The document retrieval step in RAG allows for more efficient matching of supporting documents, leading to generation of more relevant answer candidates. For instance, the clue *"Warehouse abbr."* results in *"pkg"* and *"bldg"* candidates among RAG predictions, whereas BART generates abstract and largely irrelevant strings.

Our manual inspection of model predictions suggest that both BART and RAG correctly infer the grammatical form of the answer from the formulation of the clue. For example, the clue *"Stitched"* produces the candidate answers *"Sewn"* and *"Made"*, and the clue *"Word repeated after "Que""* triggers mostly Spanish and French genera-

Model	Fact.	Hist.	Meaning	Syn./Ant.	Blank	Abbr.	Pref./Suf.	Wordplay	X-lingual	Dependent
BART	40.4	19.0	43.9	40.3	36.0	42.9	20.0	33.5	40.0	0.0
RAG-wiki	53.9	28.6	55.3	46.6	60.0	60.0	60.0	43.9	60.0	11.8
RAG-dict	54.2	35.7	52.8	48.9	61.3	85.7	60.0	46.3	40.0	11.8

Table 4: Exact match, top-20. should Omit the dependent category

tions (e.g. *"Avec"* or *"Sera"*).

As previously stated RAG-wiki and RAG-dict largely agree with each other with respect to the ground truth answers. We qualitatively assessed instances where either RAG-wiki or RAG-dict predict the answer correctly in Appendix A.

7 Discussion and Future Work

The presented task is challenging to approach in an end-to-end model fashion. There are several reasons for this, which we discuss below.

Character-level outputs. Commonly used Transformer decoders do not produce character-level outputs and produce BPE and wordpieces instead, which creates a problem for a potential end-to-end neural crossword solver. One possible solution can be the modification of the loss term, designed with character-based output logits instead of BPE since the crossword grid constraints are at a single cell- (i.e. character-) level. There is some work done in the character-level output transformer encoders such as Ma et al. (2020). However, to our best knowledge there is no major generative Transformer architecture which supports character-level outputs yet, we intend to explore this avenue further in future work to develop an end-to-end neural crossword solver.

SMT solver constraints. As mentioned earlier, our current baseline solver does not allow partial solutions, and we rely on pre-filtering using the oracle from the ground-truth answers. Although this strategy is flawed for the obvious use of the oracle, the alternatives are currently either computationally intractable or too lossy. One such strategy is to remove k clues at a time, starting with $k = 1$ and progressively increasing the number of clues removed until the remaining relaxed puzzle can be solved – which has the complexity of $O(2^n)$, where n is the total number of clues in the puzzle.Another approach we tried was to relax certain constraints of the puzzle grid, maximally satisfying as many constraints as possible, which is formally known as the maximal satisfaction problem (MAX-SAT). This is a NP-hard problem for which it is hard to find approximate solutions (Papadimitriou, 1994).

Our initial foray into such approximate solvers (Previti and Marques-Silva, 2013; Liffiton and Malik, 2013) produced severely under-constrained puzzles with garbage character entries. Further work needs to be done to extend this solver to handle partial solutions elegantly without the need for an oracle, this could be addressed with probabilistic and weighted constraint satisfaction solvers, in line with the work by Littman et al. (2002); Keim et al. (1999) and Ginsberg (2011), but without the dependency on the past crossword clues.

8 Conclusion

We present a new challenging task of solving crossword puzzles and present the New York Times Crosswords Dataset, which can be approached at a QA-like level of individual clue-answer pairs, or at the level of an entire puzzle, with imposed answer interdependency constraints. This new benchmark contains a broad range of clue types that require diverse reasoning components. We carry out a set of baseline experiments that indicate the overall difficulty of this task for the current systems, including retrieval-augmented SOTA models for open-domain question answering. We also discuss the technical challenges in building a crossword solver and obtaining partial solutions as well as in the design of end-to-end systems for this task. We hope that the NYT Crosswords task would define a new high bar for the AI systems.

9 Ethical Considerations

The New York Times daily crossword puzzles are a copyright of the New York Times. We have obtained preliminary approval from the New York Times to release this data under a non-commercial and research use license, and are in the process of finalizing the exact licensing terms and redistribution channels with their legal department. We also got permission from Parth Parikh through personal communication to modify and reuse parts of their crossword solver[6].

References

Jonathan Berant, Andrew Chou, Roy Frostig, and Percy Liang. 2013. Semantic parsing on freebase from question-answer pairs. In *Proceedings of the 2013 conference on empirical methods in natural language processing*, pages 1533–1544.

Antoine Bordes, Nicolas Usunier, Sumit Chopra, and Jason Weston. 2015. Large-scale simple question answering with memory networks. *arXiv preprint arXiv:1506.02075*.

Leonardo de Moura and Nikolaj Bjørner. 2008. Z3: An efficient smt solver. In *Tools and Algorithms for the Construction and Analysis of Systems*, pages 337–340, Berlin, Heidelberg. Springer Berlin Heidelberg.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186.

Dheeru Dua, Ananth Gottumukkala, Alon Talmor, Sameer Singh, and Matt Gardner. 2019. Orb: An open reading benchmark for comprehensive evaluation of machine reading comprehension. In *EMNLP 2019 MRQA Workshop*, page 147.

Angela Fan, Yacine Jernite, Ethan Perez, David Grangier, Jason Weston, and Michael Auli. 2019. ELI5: Long form question answering. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 3558–3567, Florence, Italy. Association for Computational Linguistics.

Matthew L Ginsberg. 2011. Dr. fill: Crosswords and an implemented solver for singly weighted csps. *Journal of Artificial Intelligence Research*, 42:851–886.

Di Jin, Zhijing Jin, Joey Tianyi Zhou, and Peter Szolovits. 2020. Is bert really robust? a strong baseline for natural language attack on text classification and entailment. In *Proceedings of the AAAI conference on artificial intelligence*, volume 34, pages 8018–8025.

Vladimir Karpukhin, Barlas Oguz, Sewon Min, Patrick Lewis, Ledell Wu, Sergey Edunov, Danqi Chen, and Wen-tau Yih. 2020. Dense passage retrieval for open-domain question answering. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 6769–6781, Online. Association for Computational Linguistics.

Greg A. Keim, Noam M. Shazeer, Michael L. Littman, Sushant Agarwal, Catherine M. Cheves, Joseph Fitzgerald, Jason Grosland, Fan Jiang, Shannon Pollard, and Karl Weinmeister. 1999. Proverb: The probabilistic cruciverbalist. In *Proceedings of the Sixteenth National Conference on Artificial Intelligence and the Eleventh Innovative Applications of Artificial Intelligence Conference Innovative Applications of Artificial Intelligence*, AAAI '99/IAAI '99, page 710–717, USA. American Association for Artificial Intelligence.

Daniel Khashabi, Snigdha Chaturvedi, Michael Roth, Shyam Upadhyay, and Dan Roth. 2018. Looking beyond the surface: A challenge set for reading comprehension over multiple sentences. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 252–262, New Orleans, Louisiana. Association for Computational Linguistics.

Tom Kwiatkowski, Jennimaria Palomaki, Olivia Redfield, Michael Collins, Ankur Parikh, Chris Alberti, Danielle Epstein, Illia Polosukhin, Jacob Devlin, Kenton Lee, et al. 2019a. Natural questions: A benchmark for question answering research. *Transactions of the Association for Computational Linguistics*, 7:453–466.

Tom Kwiatkowski, Jennimaria Palomaki, Olivia Redfield, Michael Collins, Ankur Parikh, Chris Alberti, Danielle Epstein, Illia Polosukhin, Matthew Kelcey, Jacob Devlin, Kenton Lee, Kristina N. Toutanova, Llion Jones, Ming-Wei Chang, Andrew Dai, Jakob Uszkoreit, Quoc Le, and Slav Petrov. 2019b. Natural questions: a benchmark for question answering research. *Transactions of the Association of Computational Linguistics*.

Kenton Lee, Ming-Wei Chang, and Kristina Toutanova. 2019. Latent retrieval for weakly supervised open domain question answering. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 6086–6096, Florence, Italy. Association for Computational Linguistics.

Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Ves Stoyanov, and Luke Zettlemoyer. 2019. Bart: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. *arXiv preprint arXiv:1910.13461*.

Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, Sebastian Riedel, and Douwe Kiela. 2020. Retrieval-augmented generation for knowledge-intensive nlp tasks. In *Advances in Neural Information Processing Systems*, volume 33, pages 9459–9474. Curran Associates, Inc.

Mark H Liffiton and Ammar Malik. 2013. Enumerating infeasibility: Finding multiple muses quickly. In *International Conference on Integration of Constraint Programming, Artificial Intelligence, and Operations Research*, pages 160–175. Springer.

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

Wentao Ma, Yiming Cui, Chenglei Si, Ting Liu, Shijin Wang, and Guoping Hu. 2020. CharBERT: Character-aware pre-trained language model. In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 39–50, Barcelona, Spain (Online). International Committee on Computational Linguistics.

Joshua Maynez, Shashi Narayan, Bernd Bohnet, and Ryan McDonald. 2020. On faithfulness and factuality in abstractive summarization. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 1906–1919, Online. Association for Computational Linguistics.

Tom McCoy, Ellie Pavlick, and Tal Linzen. 2019. Right for the Wrong Reasons: Diagnosing Syntactic Heuristics in Natural Language Inference. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 3428–3448, Florence, Italy. Association for Computational Linguistics.

Anav Mehta. 2021. Reinforcement learning for constraint satisfaction game agents (15-puzzle, minesweeper, 2048, and sudoku). *arXiv preprint arXiv:2102.06019*.

Timothy Niven and Hung-Yu Kao. 2019. Probing neural network comprehension of natural language arguments. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4658–4664.

Rasmus Berg Palm, Ulrich Paquet, and Ole Winther. 2017. Recurrent relational networks. *arXiv preprint arXiv:1711.08028*.

Christos H. Papadimitriou. 1994. *Computational complexity*. Addison-Wesley.

Alessandro Previti and Joao Marques-Silva. 2013. Partial mus enumeration. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 27.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. 2019. Exploring the limits of transfer learning with a unified text-to-text transformer. *arXiv preprint arXiv:1910.10683*.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. Squad: 100,000+ questions for machine comprehension of text. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2383–2392.

Anna Rogers, Matt Gardner, and Isabelle Augenstein. 2021. QA dataset explosion: A taxonomy of NLP resources for question answering and reading comprehension. *CoRR*, abs/2107.12708.

Kurt Shuster, Spencer Poff, Moya Chen, Douwe Kiela, and Jason Weston. 2021. Retrieval augmentation reduces hallucination in conversation. *CoRR*, abs/2104.07567.

Chenglei Si, Shuohang Wang, Min-Yen Kan, and Jing Jiang. 2019. What does BERT learn from multiple-choice reading comprehension datasets? *arXiv preprint arXiv:1910.12391*.

Helmut Simonis. 2005. Sudoku as a constraint problem. In *CP Workshop on modeling and reformulating Constraint Satisfaction Problems*, volume 12, pages 13–27. Citeseer.

Saku Sugawara, Pontus Stenetorp, Kentaro Inui, and Akiko Aizawa. 2020. Assessing the benchmarking capacity of machine reading comprehension datasets. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 8918–8927.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008.

Eric Wallace, Shi Feng, Nikhil Kandpal, Matt Gardner, and Sameer Singh. 2019. Universal adversarial triggers for attacking and analyzing nlp. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 2153–2162.

Yi Yang, Wen-tau Yih, and Christopher Meek. 2015. Wikiqa: A challenge dataset for open-domain question answering. In *Proceedings of the 2015 conference on empirical methods in natural language processing*, pages 2013–2018.

Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William Cohen, Ruslan Salakhutdinov, and Christopher D Manning. 2018. Hotpotqa: A dataset for diverse, explainable multi-hop question answering. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2369–2380.

Dani Yogatama, Cyprien de Masson d'Autume, Jerome Connor, Tomas Kocisky, Mike Chrzanowski, Lingpeng Kong, Angeliki Lazaridou, Wang Ling, Lei Yu, Chris Dyer, et al. 2019. Learning and evaluating general linguistic intelligence. *arXiv preprint arXiv:1901.11373*.

Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. 2019. HellaSwag: Can a Machine Really Finish Your Sentence? In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4791–4800.

Sheng Zhang, Xiaodong Liu, Jingjing Liu, Jianfeng Gao, Kevin Duh, and Benjamin Van Durme. 2018. Record: Bridging the gap between human and machine commonsense reading comprehension. *arXiv preprint arXiv:1810.12885*.

A Qualitative Analysis of RAG-wiki and RAG-dict Predictions

We additionally examined the top-20 exact-match predictions generated by RAG-wiki and RAG-dict. With some exceptions, both models predict similar results (in terms of answer matches) for around 85% of the test set. We further analyzed the instances where the models behaved differently.

Table 5 shows examples where RAG-dict failed to generate the correct predictions but RAG-wiki succeeded, and vice-versa. Most of the instances where RAG-dict predicted correctly and RAG-wiki did not are the ones where the target closely related to the meaning of the source. The instances where RAG-wiki predicted the exact match but RAG-dict couldn't are the examples where the target is not a direct meaning of the source, however some more information around the source is good enough to predict the target. For Historical category, both the models could not predict correctly when the source length was more than four words. The models didn't have much variation in the results for rest of the other categories.

Category	RAG-dict predicts correctly,RAG-wiki fails	RAG-wiki predicts correctly, RAG-dict fails
Factual	Source: “Asian nursemaid”, Target: “amah”. Source: “Pill alternative, for short”, Target: “iud”	Source: “Quisling’s city”, Target: “oslo”. Source: “ Avatar of Vishnu”, Target: ”rama”
Word Meaning	Source: “Pause indicator”, Target: “comma”. Source: “Moves along quickly”, Target:” scoots”	Source: “Sites for grand entrances”, Target: “archways”. Source: “Point of no return?”, Target: “ace”.
Word Play	Source: “Kind of contribution”, Target: ”ira”. Source: “Without ice”, Target:”neat”	Source: “I’m impressed!”, Target: “ooh”. Source: “Airport no no , Target: “ knife”.
Synonyms	Source: “Stitched”, Target: ”sewn”. Source:	Source: “guess “ , Target: “ idea”.
Antonyms	“Promptly”, Target: “on time”.	
Fill in the Blanks	Source: “__rug “, Target: “ area”. Source: “canola __ “, Target: “ oil”.	Source: ”__-Israeli relations”, Target : ”arab”.

Table 5: Examples where one of the models: RAG-dict, RAG-wiki predicts correctly and other fails. Examples are for exact match, top-20