Automated crossword solver using GPT-4 and logic constraints

Hillary Tao, School of Computer Science
McGill University, Montreal
March, 2024

A thesis submitted to McGill University in partial fulfillment of the requirements of the degree of

Master of Science

Abstract

With extraordinary advancements in Natural Language Processing (NLP) and crosswords being more popular than ever, the current era of Machine Learning (ML) and NLP has reinvigorated interest in automating crossword solving. Crossword puzzles are intellectual games that are popular all over the world. To solve them, they require a large vocabulary and deductive reasoning. Crossword clues are often less straightforward, involve many different types of reasoning, and could contain multiple different linguistic complexities. Furthermore, they are often considered an ideal testing ground for AI and NLP tasks as they are an abundant, diverse and validated type of data. In fact, they are often used for QA and constraint satisfaction. However, solving them involves more than just generating answers. It is essential to take into account intersecting letters and length constraints. Nevertheless, crossword solvers often encounter the following two problems: a vast search space needed to generate candidate solutions and having multiple valid possible solutions. Therefore, developing models capable of accurately interpreting and resolving linguistic ambiguity in crossword clues can help contribute to advancements in NLP, particularly in tasks that involve deciphering natural language. As previously mentioned, despite advancements, challenges persist. Automated crossword solvers have been built in the past and can outperform most humans. Some of the best are Proverb, Dr. Fill, and Berkeley Solver. Despite their success, they still struggle to solve difficult linguistic instances present in crosswords and fail to outperform expert humans. In pursuit of improving crossword-solving accuracy, previous models using BERT, T5, and GPT2 still faced difficulties when it came to QA due to clue ambiguity and crossword grid constraints. By leveraging GPT-4 alongside the findings of past explorations in this field, the goal of this thesis is to develop a cutting-edge crossword solver that achieves higher

accuracy than previous models and potentially address the remaining challenges in crossword solving, making our approach a promising avenue for further improvement in the field. To achieve this, we developed a crossword solver model incorporating GPT-4 for the generation of answers along with grid constraints to take care of the crossing letter and length constraints. We tested our work using a collection of crosswords extracted from the New York Times. Our experiments show that our model achieved an accuracy that is on more than 20% higher than the latest crossword solver presented in at the 2022 ACL conference and about 2% higher than the Berkeley Solver.

Abrégé

Avec le New York Times Crosswords qui a atteint plus de 500 000 abonnés en 2019, l'ère actuelle de l'apprentissage automatique (ML) et du traitement du langage naturel (NLP) a a renouvelé l'intérêt pour l'automatisation de la résolution de mots croisés. Les mots croisés, populaires dans le monde entier, sont des défis intellectuels qui nécessitent un vocabulaire étendu et le raisonnement déductif. Les indices donnés sont souvent moins simples que l'on pense. Ils impliquent divers types de raisonnement et des complexités linguistiques. En tant que tels, ces énigmes constituent un terrain parfait pour tester les tâches de AI et de NLP, en particulier la réponse aux questions (AQ) et la satisfaction des contraintes. La résolution implique plus que de produire des réponses. Les solveurs de contraintes sont essentiels pour réconcilier les lettres de croisement et contraintes de longueur. Ceci est difficile en raison de devoir chercher des candidats dans un grand espace et de la d'avoir plusieurs sources valides. solutions. Par conséquent, le développement de modèles capables d'interpréter et de résoudre avec précision l'ambiguïté linguistique dans les indices de mots croisés peuvent contribuer aux progrès de la PNL, en particulier dans les tâches qui impliquent comprendre et lever l'ambiguïté du langage naturel. Comme mentionné précédemment, malgré les progrès, les défis persistent. Des solutions automatisées de mots croisés ont été créées dans le passé et peuvent surpasser la plupart des humains. Certains les meilleurs sont Proverb, Dr. Fill et Berkeley Solver. Malgré leur succès, ils ont encore du mal à résoudre des difficultés linguistiques présentes dans les mots croisés et ne parviennent pas à surpasser les humains experts. Dans le but d'améliorer la précision de la résolution des mots croisés, les modèles précédents utilisant BERT, T5 et GPT2 avaient toujours des problèmes losqu'il s'agissait de la compréhension de l'ambiguïté des indices et des constraintes imposées par la grille des mots croisés. En

tirant parti de GPT-4 parallèlement aux découvertes du passé dans ce domaine, l'objectif est de développer un outil de résolution de mots croisés de pointe qui atteint des résultats plus élevés que les modèles a priori et potentiellement résoudre les défis restants dans la résolution de mots croisés. Pour y parvenir, nous avons développé un modèle utilisant GPT-4 pour la génération de réponses ainsi que les solveurs de la théorie du module de satisfaction (SMT) pour prendre en charge les contraintes de lettre et de longueur de croisement. Nous avons testé notre solution sur une collection de mots croisés extraits du New York Times. Nos expériences montrent que notre modèle a atteint une précision supérieure de plus de 20% que celle du dernier solveur de mots croisés présenté lors de la conférence ACL 2022 et environ 2% supérieure à celle du solveur de Berkeley.

Acknowledgements

I would like to thank my supervisor, Professor Xue Liu, for his invaluable guidance, encouragement, and patience throughout my entire study and thesis writing. His knowledge and help were essential for the completion of my thesis.

I would also like to thank Zhaoyu Li for his contribution to this thesis. His feedback and ideas for experiments were essential in this research.

Acronyms

NLP: Natural Language Processing

ML: Machine Learning

API: Application Programming Interface

CSP: Constraint Satisfaction Problem

SAT: Boolean Satisfiability Problem

SMT: Satisfiability Modulo Theory

LLM: Large Language Model

NYT: New York Times

GPT: Generative Pre-trained Technology

BERT: Bidirectional Encoder Representations from Transformers

T5: Text-to-Text Transfer Transformer

CNN: Convolutional Neural Network

RNN: Recurrent Neural Network

QA: Question-Answering

CDCL: Conflict-Driven Clause Learning

DPLL: Davis–Putnam–Logemann–Loveland

CNF: Conjunctive Normal Form

NP: Nondeterministic Polynomial

ReLU: Rectified Linear Unit

API: Application Programming Interface

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Problem Statement

With a rapid increase in the performance of NLP models in recent years, it reinvigorated researchers' interest in NLP models' capabilities to reason about popular word games. In the realm of mind games, crossword puzzles have emerged as a crowd favourite.

Popular all over the world, crossword puzzles are challenges that require an expansive vocabulary, as well as deductive reasoning. With more than 500,000 subscribers in 2019 for the NYT crosswords [1], the current era of ML and NLP has also contributed to the renewed interest in automating crossword solving.

They offer an abundant amount of diverse labeled data. As such, these puzzles serve as a perfect testing ground for AI and NLP tasks, specifically question answering (QA) and constraint satisfaction.

Solving crosswords requires more than just generating words that fit the grid. One needs to decipher clues, which may include puns, anagrams, or metaphorical associations. These clues are often less straightforward and involve various types of linguistic complexities, such as homophony, and wordplay, to name a few [17, 32]. Furthermore, crossword clues can sometimes intentionally lack specificity, requiring those involved to skip certain clues until more information is available [17, 20, 32].

Some of the best automated crossword solvers in the world are Proverb [20], Dr. Fill [17], and Berkeley Solver [32]. Despite their success, they still struggle to understand difficult linguistic nuances present in crosswords and fail to outperform expert humans.

The main problem with current solvers is their struggle with addressing the complex linguistic nuances of clues and the rigid structural constraints of the crossword grid simultaneously. Generally, existing solvers excel in one of the above aspects, but often struggle with the other.

Current solvers, which rely heavily on mathematical optimization or rule-based approaches, can struggle to interpret subtle linguistic cues accurately. This leads to a mismatch between the solver's interpretation and the intended answer, resulting in inaccurate solutions. This is in part due to the vast search space and potential for multiple valid solutions. Miscalibrations in models might also result in predicted answers that are close to the actual solutions but still wrong [32]. Addressing these challenges necessitates a solver capable of interpreting linguistic subtleties, generating contextually more correct solutions.

1.2 Thesis Methodology and Contribution

To improve on existing solvers, this thesis' methodology consists of developing an automated crossword solver that employs GPT-4 for the generation of answers and logical constraints to take care of the intersecting letters and length constraints. To validate the model's efficacy, it was tested on an extensive dataset comprised of crossword puzzles from The New York Times (NYT) ranging from 1989 to 2018 and comparing its efficacy to previous models.

I worked on both the theoretical conception and the actual coding portion of the model. As well, I prepped the data used for the testing aspect. Finally, I wrote the entirety of my thesis.

1.3 Research Objectives

By leveraging the latest GPT model alongside the findings of past explorations in this field, the goal is to develop a crossword solver that achieves higher accuracy than previous models and potentially addresses the current challenges in crossword solving. As such, it highlights our approach as a promising avenue for further improvement in the field.

1.4 Outline of the Thesis

The thesis is organized as follows:

- Chapter 1: The background information and context for this thesis are provided in this chapter. It delivers an overview of the thesis' organization and presents the research technique and objectives.
- Chapter 2: Background and Related Work- This chapter contains a short history on crosswords and crossword solvers, reviews the relevant literature and provides a theoretical foundation for the implementation of crossword solvers.
- Chapter 3: Models and Solutions - This chapter discusses the design of the final crossword solver model in detail as well as different techniques that we considered.
- Chapter 4: Experiments and Evaluations- The experimental setup and technique utilized to assess the performance of our final crossword solver model are presented in this chapter. It also contains the outcomes.
- Chapter 5: Discussion - This chapter dives into detail the results of our experiments, where we look into the advantages and limitations of our final model.
- Chapter 6: Conclusion - The summary of our results and the project's contribution are compiled in this chapter. It also includes a list of potential future projects.

Chapter 2

Literature Review

In Chapter 2, we review some of the most pertinent work in relation to this thesis, starting with a brief history on crosswords and its solvers, then moving on to CSPs, LLMs and finally conclude with summaries of some of the most relevant crossword solvers.

2.1 Brief History of Crosswords and Crossword Solvers

Before diving into the relevant research surrounding this thesis, we first set the stage with a brief history on crosswords and crossword solvers.

The history of crossword solvers traces its origins back to the late 19th century, where in 1913, Arthur Wynne created the first known crossword puzzle for the New York World [28]. Since it inception, a crossword puzzle, as displayed in 2.1, is a word game or mind-teaser that typically consists of the following several key elements [3,28]:

- Grid: The grid is the main structure of the crossword, consisting of black and white squares. The white squares are to be filled with letters to form the correct words, while black squares are there to separate and delineate the words.

- Clues: Clues are hints provided for each word in the crossword. They can take on various forms such as definitions, wordplay, or references to certain topics.

- Across and Down: Words in a crossword are generally categorized as either ”across” or ”down”. ”Across” words are read from left to right, and ”down” words are read from top to bottom. They can sometimes also be referred to as ”horizontal” or ”vertical” respectively.

- Numbers: Each white square contains a number which indicates the starting point of an across or down word. The numbers help solvers identify the beginning of each word and link the clues to the appropriate grid location.

Figure 2.1: Example of a Crossword taken from the New York Times Journal [1]

In the 20th century, crosswords have crossed into the realm of computational problem-solving, where automated crossword solvers have flourished. Notable ones include Proverb [20], Dr.Fill [17], and Berkeley Solver [32], demonstrating the capability to outperform a majority of humans by combining logic constraints and LLMs.

To better understand crossword solvers, we will be exploring CSPs, SATs, LLMs and notable crossword solvers as we dive further into this thesis.

2.2 Constraint Solving Problems

Crossword solvers are often challenged by grid constraints and possess multiple plausible candidate answers. Participants must ensure that their answers satisfy both the grid constraints and the given clues. As such, they are a great example of CSPs. In this section, we will be looking into the brief history of CSPs, their definition and key elements, their sub-branches, and finally some of their most popular tools.

A CSP is a mathematical element employed in computer science and AI to model problems by using a set of variables whose values are subjected to specific constraints [11]. CSPs can be found in numerous applications, including scheduling, planning, resource allocation, configuration, and optimization, to name a few.

In a CSP, the primary objective is to determine an assignment of values to the variables such that it satisfies all imposed constraints [11, 18, 30]. Each variable has an associated domain, which defines the possible values it can have. Constraints, on the other hand, refer to the relationships and dependencies between variables, which consequently also dictate how these variables relate to one another.

The solution of a CSP is a combination of variable assignments that adheres to the problem's defined constraints. However, it is not as simple as it sounds. The challenge lies in searching through a large solution space efficiently. Techniques such as backtracking, constraint propagation, and heuristics are often employed to minimize this problem [30].

Additionally, CSPs allow for the representation and solution of complex, real-world problems in a structured format. Their inherent flexibility and applicability make CSPs an important factor in computational techniques to address a wide array of problems requiring logical reasoning and constraint-based decision-making.

As previously mentioned, in the following subsections, we will be going through their brief history, definition, key elements and some of the most important work and research areas to have come out of it.

2.2.1 Brief History

CSPs found their roots in the mid-20th century. Throughout that time, the complexities of scheduling, resource allocation, and logistical challenges pushed mathematicians to find ways to deal with these problems in an efficient manner.

However, CSPs were only really formalized in the 1970s and 1980s. Researchers came up with systematic methodologies for the representation and resolution of problems subject to constraints. During this time, important contributions included the backtracking algorithm, along with the introduction of constraint propagation techniques, were released [30].

The subsequent decades saw the use of constraint satisfaction techniques in AI programs [30]. CSPs assumed a pivotal role in the modeling and resolution of intricate challenges pertaining to knowledge representation, planning, and decision-making within AI research and application [30].

The start of the 21st century ushered in a period of time characterized by an increasing use of constraint solving methodologies with more AI programs. Noteworthy applications of constraint programming during this era include scheduling, configuration, and optimization. [18, 30].

Current trends in constraint solving encompass the deployment of sophisticated algorithms to harness the potential of parallel computing programs and automated reasoning. Constraint solving techniques continue to be a crucial part of addressing complex problems in domains such as robotics, bioinformatics, and supply chain management to name a few [13].

2.2.2 Definition and Key Elements

To better understand a CSP, a definition and its key elements will be provided below.

A CSP is commonly described as a method used to model and solve problems where a set of variables must be assigned values subject to specified constraints. Its primary goal is to find a consistent assignment of values to the variables that satisfies all the constraints [11, 30].

A CSP typically contains the following:

- Variables (V): Variables represent the unknowns whose values need to be determined in the problem. They can be any value from a specified domain, which defines the set of possible values it can assume.

- Domains (D): The domain of a variable is the set of values it can take on. Depending on the nature of the problem, they can be finite or infinite, as well as discrete or continuous. A problem's constraints ensure that the assigned values come from these domains.

- Constraints (C): Constraints are rules that impact the allowable combinations of values for a set of variables. They dictate relationships or dependencies among variables and must be satisfied for a solution to be valid.

- Solution (S): A valid solution to a CSP is an assignment of values to the variables that satisfies all the constraints. The search for a solution often involves systematically exploring the space of possible assignments until a satisfactory set of values is found.

- Objective Function (optional): In some CSPs, there may be an associated objective function that needs to be optimized. The objective function assigns a value to each possible assignment, and the goal is to find the assignment that optimizes or satisfies this function.

- Search Space: The search space of a CSP is the set of all possible assignments of values to variables. Due to the constraints, not all combinations are valid solutions,

and the search process involves going through this space to find a consistent assignment [15].

CSPs are often solved using search algorithms, with backtracking being one of the fundamental techniques used. Backtracking involves systematically exploring the search space and undoing choices when a constraint is violated. Other search algorithms, like constraint propagation or heuristics, may also be employed to efficiently traverse the space.

In summary, a CSP consists of a set of variables with associated domains, a set of constraints that define allowable combinations of variable assignments, and the objective of finding a consistent assignment satisfying all constraints. The exploration of the search space is a key aspect of solving CSPs.

2.2.3 Boolean Satisfiability Problems

In many cases, CSPs are reformulated into SATs for the purpose of leveraging the efficiency of SAT solvers. This reformulation is motivated by the fact that SAT solvers have contributed to substantial advancements and optimizations in AI and other areas, making them powerful tools for solving complex logical problems.

SATs form a crucial component in the domain of mathematical logic and computational complexity. Based on the theoretical framework of propositional logic, SATs can be defined by the quest to determine the feasibility of truth assignments to a set of Boolean variables that satisfy a given logical formula [16].

SATs are typically expressed in CNF, where logical relationships among variables are structured as clauses. The following figure illustrates an example of a SAT problem [15].

With the importance of SAT, the introduction of efficient SAT solvers was inevitable, where employing techniques like backtracking, conflict-driven clause learning, and advanced heuristics, has propelled the applicability of SATs across diverse fields, including formal verification, AI, and hardware design for instance [22].

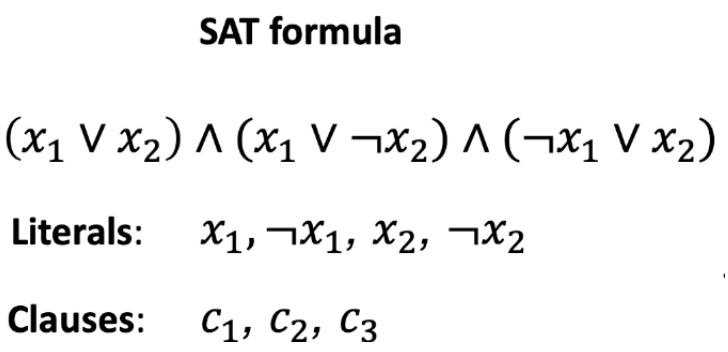

Figure 2.2: Example of a SAT Problem

2.2.4 SAT Solvers

SATs and their solvers are some of the foundational components of computational logic. The motivation behind their development lies in the complexity of a SAT problem, an important NP-complete problem that has spurred the creation of powerful tools for solving intricate logical decision problems.

Driven by the desire to efficiently explore the Boolean solution space, SAT solvers have evolved quickly over time, resulting in the development of ground-breaking applications like CDCL [29]. These solvers employ strategies such as backtracking to methodically search through the solution space, which is frequently represented in CNF, and arrive at the best answer. Guided by heuristic insights, the fundamental idea of SAT solvers is to adeptly explore the large solution space by making strategic judgments regarding truth assignments in relation to Boolean variables.

The first step in most SAT solvers is to encode a given problem into its respective CNF representation, which expresses logical relationships in the form of conjunctions of clauses. Subsequently, the solver proceeds to methodically investigate potential truth assignments, traversing several branches of the solution space via backtracking. Furthermore, sophisticated heuristics can also help direct variable selection and truth value assignment order, maximizing the effectiveness of the search.

The CDCL algorithm is one of the most significant advancements in constraint satisfaction. It is a crucial part of SAT solvers, enabling them to dynamically modify their search strategy in response to conflicts that they encounter [29]. In order to prevent re-

occurring conflicts, the solver learns from conflicts by recognizing and evaluating the conflicting clause and then modifying its approach.

Although SAT solvers have demonstrated their effectiveness in several applications, they are not without their drawbacks. For example, its NP-completeness-related computational complexity makes scaling difficult. As well, their expressiveness is restricted to Boolean logic, which limits the kinds of constraints they can handle well [16, 22].

Ongoing research continuously refines SAT-solving methodologies, contributing to their versatility and significance in addressing complex logical decision problems. The perpetual exchange between theoretical insights and algorithmic advancements characterizes the academic landscape surrounding SATs.

2.2.5 SMT

SMT solvers have only relatively recently emerged as powerful tools in automated reasoning. It solves decision problems by integrating mathematical theories with logical restrictions [14]. Building upon the foundations laid by SAT solvers, SMT solvers combine specialized decision processes for certain mathematical theories, including arrays, bit-vectors, and arithmetic. With applications ranging from formal verification, software analysis, and symbolic execution, this allows users to reason about complicated systems that have both logical and mathematical components [4, 14].

The history of SMT solvers can be traced back to the early 2000s, with pivotal research such as the development of the DPLL framework [**?**] and the introduction of efficient decision procedures for theories like linear arithmetic and bit-vectors. Over the years, research efforts have focused on refining algorithms, optimizing performance, and expanding the expressiveness of SMT solvers to handle increasingly complex problem domains.

Compared to SAT solvers, SMT solvers offer several other advantages. Firstly, they provide a more expressive modeling language, allowing users to illustrate a broader range of constraints [4, 14]. This enables SMT solvers to capture richer problem domains and reason about complex systems more accurately. Additionally, SMT solvers often ex-

hibit better performance on problems that involve both logical and mathematical constraints, as they can leverage specialized decision procedures tailored to specific theories.

Despite their strengths, SMT solvers also have their limits. One weakness is the computational complexity of solving problems in certain areas, such as nonlinear arithmetic or quantified formulas. These problems can be challenging to solve efficiently, limiting the scalability of SMT solvers for certain classes of problems. Furthermore, while SMT solvers excel at reasoning about precise mathematical constraints, they may struggle with problems that involve uncertainty, approximation, or probabilistic reasoning [4, 14].

SMT solvers represent a significant advancement in automated reasoning, offering a versatile framework for solving decision problems that combine logical and mathematical constraints. Their integration of specialized decision procedures, expressive modeling language, and efficient algorithms has propelled them to the forefront of formal methods research and practical applications. However, addressing challenges such as scalability and handling uncertainty remains an ongoing area of research for those involved.

2.2.6 Z3-Solver

One of the most well-known SMT solvers is the Z3 solver. The Z3 solver is a powerful and popular SMT solver developed by Microsoft Research. It is primarily used for solving constraints from various domains, including software analysis, verification, and security protocols [10].

The main motivation behind Z3 is to provide a highly efficient solver for a wide range of constraint satisfaction problems (CSPs). These problems arise in various domains, including software and hardware verification, model checking, and scheduling to name a few. It aims to automate the process of proving or disproving the satisfiability of logical formulas and constraints [10].

While Z3 is highly efficient for many types of constraints, certain problems can still be computationally expensive or even undecidable [10]. Constraints involving non-linear arithmetic, quantifiers, or complex data structures may require significant computational resources or may not even be solvable at all. Although it supports a wide range of theo-

ries and logical constructs, there are still limitations to its expressiveness. Certain specialized domains or problem types could require custom solvers or additional pre-processing steps to effectively utilize Z3 [10].

Furthermore, solving complex constraints can consume significant computational resources, including memory and CPU. As a result, scalability can be another downside, particularly for large-scale verification tasks or when dealing with highly complex constraints.

The Z3 solver is a powerful tool for automated constraint solving and formal verification tasks, offering efficiency and support for various problem domains. However, users should be aware of its limitations regarding complexity, expressiveness, resource consumption, and the need for careful validation of results.

2.3 LLMs

Prior to looking at crossword solvers, it is important to know how they came to be. One of the most important aspects that has led to some of the best crossword solvers is the use of LLMs.

In this chapter, we will be giving a short history on LLMs, followed by some of the most important turning points in their evolution such as the introduction of the Transformers architecture, BERT, GPT, and many more.

2.3.1 Brief History

The evolution of LLMs in NLP unfolds as a series of advancements that have significantly impacted the progression of ML. In 2018, the introduction of Google's BERT [12] represented a pivotal moment, enhancing language comprehension by capturing contextual nuances bidirectionally. Following suite, 2019 witnessed the debut of T5, a model that revolutionized NLP tasks by framing them as text-to-text challenges, streamlining versatility and training procedures.

One of the most recent milestones was in 2020 with the release of GPT-3 by OpenAI. GPT-3 showcased remarkable language generation capabilities across a multitude of tasks, from sentence completion to language translation to name a few. At the same time, models like DistilBERT and RoBERTa emerged, aiming to improve the efficiency of large models.

At the core of all of these models lies the transformer architecture, introduced by Vaswani et al. in 2017 [31], which played a fundamental role in capturing intricate language patterns through its self-attention mechanism.

2.3.2 Transformers Architecture

Motivated by the limitations of traditional sequential models, such as RNNs, the Transformer architecture [31], introduced by Vaswani et al. in 2017, became a turning point in the realm of NLP which subsequently inspired other revolutionary models such as BERT and GPT.

At its core, the Transformers architecture embodies a novel approach to sequence processing. RNNs struggled to capture dependencies over long sequences due to vanishing and exploding gradient problems. As well, they processed sequences sequentially, limiting parallelization [26,31]. This inefficiency hindered the training of large models on vast datasets. Furthermore, along with CNNs, they lacked a built-in mechanism to consider the positional information of elements in a sequence which is essential for tasks involving ordered data such as natural language. The motivation for the creation of the Transformer was to overcome the limitations of sequential models, enabling more scalable and efficient processing of language sequences [26,31].

With the Transformers architecture, it replaces sequential computation with a self-attention mechanism, allowing the model to simultaneously consider all elements in a sequence. This not only facilitates parallelized computations but also empowers the model to capture richer contextual relationships across an entire sequence [26, 31]. The architecture introduces the concept of attention "heads," working collaboratively to focus on

distinct aspects of the input, thereby enhancing the model's capacity to discern diverse features in a more comprehensive manner [31].

The introduction of positional encoding further addresses the challenge of maintaining sequential order information, ensuring that the model understands the position of each word in a sequence. To take into account non-linearity, feed-forward neural networks with ReLU activation are incorporated [31]. Additionally, layer normalization and residual connections are employed to stabilize the training of deep models.

The architecture's success in parallelization and contextual understanding has not only propelled its adoption in various NLP applications but has also influenced subsequent models across domains beyond NLP [33]. It represents a groundbreaking shift in NLP, offering a versatile and efficient solution to the challenges posed by traditional sequential models. Its impact extends beyond its initial motivation, shaping the landscape of modern ML.

2.3.3 BERT

Following the previously mentioned milestone, in 2018, BERT was introduced to the world by Devlin et al. through their research paper [12]. In their paper, they presented their BERT model as a pre-training technique for natural language understanding tasks. Its introduction marked a significant advancement in the field of NLP, particularly in understanding bidirectional context for language representation.

BERT was developed to overcome the contextual understanding problem and solve the shortcomings of conventional language models [12]. Through the realization of how much one's understanding is impacted by contextual details in natural language, which was something that models prior to BERT lacked [12], it inspired the authors [12] to create BERT. By utilizing bidirectional contextual modeling instead of the unidirectional methods used in earlier models, BERT was able to reach previously unheard-of outcomes by taking into account both left and right context during its pre-training.

When it comes to pre-training, BERT thoroughly processes large datasets, where it leverages the masked language modeling objective. By predicting masked words within

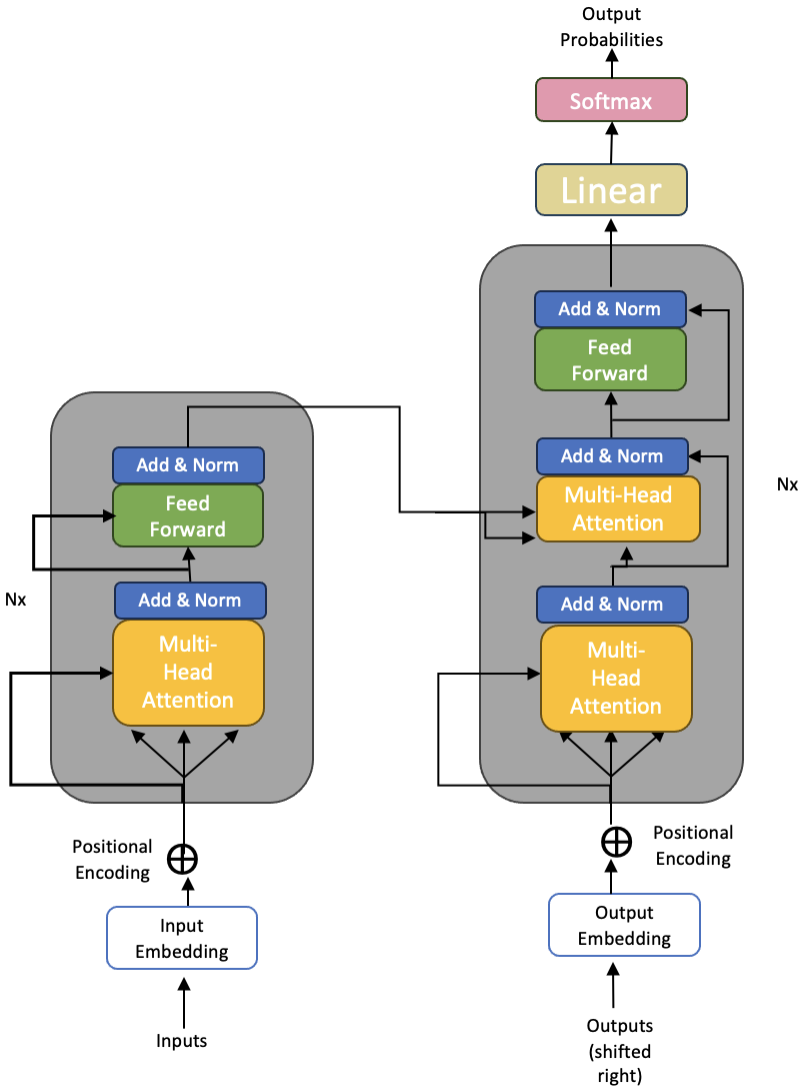

Figure 2.3: Transformer Architecture Introduced by Vaswani et al. [31]

a sentence, it captures detailed contextual dependencies and produces embeddings that condense complex relationships between words [12]. This pre-training method has allowed BERT to excel in a wide range of NLP tasks, such as sentiment analysis, named entity recognition, and question answering for instance.

The fact that BERT can provide a robust and flexible solution for a wide range of applications, beyond the limitations of task-specific designs, highlights its influence on NLP. The approach does, however, have a unique set of issues [12]. At the time of its release, it took a significant amount of money to train and fine-tune. Additionally, BERT only

works with fixed context lengths, which poses problems in situations where flexibility is required to adjust to different context lengths [12].

The seminal work of Devlin et al. has not only established BERT as a cornerstone in NLP but has also catalyzed ongoing research in later transformer-based models.

2.3.4 GPT

GPT-1, launched by OpenAI in 2018, was the precursor to many of the most advanced models that we know of today such as ChatGPT and Sora. Its initial goal was to advance the field of NLP using pre-training techniques. Its main concept is based on its ability to pick up complex linguistic patterns through its exposure to large amounts of diverse textual data during its pre-training phase [2]. GPT-1 can successfully capture local dependencies because of this design decision, which uses the transformer architecture and predicts words in a sentence based only on the previously given context [2].

The model gains a deeper understanding of natural languages' syntactic patterns, semantic connections, and contextual subtleties by pre-training it on a large variety of linguistic datasets. Its generative capabilities have allowed it to be recognized as a noteworthy turning point in NLP research through a demonstration of its capacity to create content that is both coherent and contextually relevant [2].

GPT-1 does, however, have a few significant drawbacks. Its utilization of a unidirectional context window greatly restricts its ability to handle bidirectional relationships, which consequently affects its comprehension of subtle nuances in natural language, especially in scenarios that call for a more comprehensive contextual awareness. When producing vast volumes of text, keeping a consistent logic throughout its output also proved to be difficult, occasionally leading to consistency failures [2]. Furthermore, GPT-1's susceptibility to generating factually incorrect information really emphasized the inherent complications of unsupervised learning regarding enormous and diverse datasets.

Despite its problems, GPT-1 serves as an important breakthrough, setting the stage for subsequent advancements in LLMS. The model's launch sparked a wave of innova-

tion, with each version increasing in strength. GPT-1's major role in transforming the landscape of pre-training approaches drives continuous study and discovery in NLP.

2.3.5 GPT-2

With the release of GPT-2 by OpenAI in 2019, which replaced GPT-1, LLMs underwent a radical transformation. The objective of this new iteration was to further enhance natural language generation and understanding, building on the successes of its predecessor. Similar to GPT-1, GPT-2 employs a transformer architecture, although it operates on a much larger scale [25].

The primary element of improvement is GPT-2's capacity to produce words with the backing of a richer and more complex contextual understanding. The model's extensive pre-training on a wider range of literary works enabled it to identify more intricate linguistic patterns, semantics, and contextual links. GPT-2 has significantly more sophisticated generating skills than its predecessor, allowing it to produce texts that are more aware and suitable for the given situation. As such, it is therefore very good at tasks like creative writing and summarizing [25].

On the other hand, GPT-2's size and power also provide particular difficulties. Training and fine-tuning are extremely costly due to the sheer amount of parameters, which, as a result, increases processing demands. As the model can produce extremely convincing false text, there have also been concerns expressed over the abuse of the technology in academic settings [25]. As well, it also sparked questions about the possible use of unauthorized data for its conception.

GPT-2's introduction marks a significant advancement in NLP, pushing the boundaries of what large-scale language models can achieve. Its impact resonates in both research and applications, influencing subsequent developments and inspiring a new era of exploration in this field.

2.3.6 GPT-3

GPT-3, which was first released in 2022, also makes use of a transformer design, which is renowned for its capacity to identify complex contextual relationships in sequential data. GPT-3 is significantly larger than most contemporary language models and its predecessor, GPT-2, with an incredible 175 billion parameters [8].

The architecture consists of many levels of attention processes, each of which increases the model's ability to understand and generate logical language. GPT-3 gains exposure through the use of a broad and diverse dataset during pre-training, which facilitates the understanding of intricate grammatical structures, linguistic patterns, and contextual relationships [8].

As well, GPT-3 is unique in the sense that it can be used for both zero-shot and few-shot learning [8]. With few-shot learning, the model can complete tasks with few instances, and with zeroshot learning, it can complete tasks without explicit training by using the general information it has gained prior to training. Thanks to its adaptability, GPT-3 may be used for a wide range of activities, including creative writing, code development, language translation, and summarization [8].

By making it available through an API, OpenAI encourages innovation in NLP and AI-driven solutions by allowing developers to incorporate its capabilities into a wide range of applications. However, on the other side, by making it available to the public, it raises a few concerns. As GPT-3 can produce realistic and contextually coherent language, ethical concerns about its responsible usage highlight the significance of using AI ethically [8].

It faces another prominent problem: large costs. It allows for fine-tuning, which involves adjusting the model to comprehend more specific tasks or domains. As a result, by subjecting it to datasets tailored to particular tasks, it boosts the model's performance. However, the sheer amount of information typically used for fine-tuning presents difficulties with regard to computing needs, requiring significant resources for deployment and training, which as a result, presents an exorbitantly high bill for the subscribers [8].

To summarize, the architecture of GPT-3, with its large scale and attention mechanisms, as well as its few-shot and zero-shot learning capabilities, places it at the forefront

of natural language processing and pushes the limits of what is possible in terms of language generation and understanding.

2.3.7 GPT-4

GPT-4, developed by OpenAI, is a large-scale multimodal model that debuted in 2023 and can generate text outputs from picture and text inputs [23].

The transformer-based model GPT-4 has been pre-trained to anticipate the next token in a document. The introduction of post-training alignment leads to enhanced performance on assessments of factual accuracy and compliance with intended behavior. The creation of optimization techniques and infrastructure that exhibit predictable behavior at a variety of sizes was a fundamental element of this paradigm.

In terms of limitations, similar to its previous iterations, it is very computationally expensive in terms of training and fine-tuning. As well, ethical concerns continue to persist, especially in terms of picture generation, where many artists have raised concern over the origins of the data used to train GPT-4.

GPT-4 demonstrates a human-level of performance based on a variety of professional and academic standards, surpassing its previous iterations [23]. As such, it makes the public even more excited for what is to come in NLP.

2.3.8 Chat GPT

In 2022, OpenAI created and released ChatGPT, a complex dialogue system that advanced conversational AI and drew on the achievements of previous models such as GPT-2. ChatGPT was developed as a sister product to InstructGPT and is now capable of carrying on meaningful discussions with people, demonstrating improvements in NLP, according to [8, 25]

ChatGPT uses the transformer architecture to record contextual dependencies and provide contextually relevant replies, driven by the goal of developing more dynamic and engaging conversational bots. The pre-trained model is able to comprehend con-

text changes, user inputs, and linguistic subtleties thanks to a variety of conversational datasets [8].

Even though it excels at providing contextually meaningful answers during conversations, there are still some issues. One of which is that it has been shown to be sensitive to how an input is worded, and in certain cases, it might produce answers that seem reasonable but are not factually correct. In order to overcome these issues, OpenAI gradually launched ChatGPT and improved it based on user input [23]. Another issue, similar to previous GPT models, is the concern surrounding plagiarism in relation to papers and homework solutions being artificially generated. To counter this, many others have launched GPT detection tools, however they have been met with lukewarm reactions. Sometimes it marked a real human's writing as AI and vice versa.

With the launch of ChatGPT, conversational AI has advanced significantly and shown promise for more interactive and contextually aware virtual assistants. Like any AI model, it is always being improved and adjusted in an effort to produce better chatbots that are increasingly more intelligent and reliable.

2.4 Crossword Solver

Formed by the amalgamation of LLMs and CSPs, many different crossword solvers were born. The introduction and summaries of the some of the most recognizable crossword solvers take place in the following subsections.

2.4.1 Proverb

Though it is hard to pinpoint when the first automated crossword solver appeared, one notable early example is "Proverb," a crossword-solving system introduced in 1999. Proverb was created by Michael L. Littman, Greg A. Keim, and Noam M. Shazeer [20]. Motivated by the new era in ML and NLP, by employing a combination of machine learning and NLP techniques to tackle crossword puzzles, it became one of the first notable attempts

to automate crossword solving. In particular, it was awarded the 1999 Outstanding Paper Award from AAAI.

The Proverb Crossword Solver is a specialized tool designed to assist crossword enthusiasts in solving crossword puzzles where the clues are proverbs or well-known sayings [20]. Users input the clues provided in the crossword puzzle, which typically consist of partial phrases or keywords associated with specific proverbs or sayings [20]. Next, in order to identify possible matches, the solver compares the input clues with entries in a chosen database that contains a sizable collection of proverbs, idioms, and sayings from many cultures and languages [20]. The solver then identifies possible completions or answers for the crossword puzzle clues based on the matches discovered.

However, one major limitation is that the accuracy of the solver depends on the quality and comprehensiveness of the database of proverbs it accesses. In some cases, the solver may struggle to find suitable matches for ambiguous or obscure clues [20].

Though it has its challenges, Proverb became a groundbreaking innovation that inspired others to follow suit.

2.4.2 Dr. Fill

Developed by computer scientist and entrepreneur Dr. Matt Ginsberg in collaboration with other well-established researchers, comes another ground-breaking crossword solving tool called Dr. Fill. When it comes to solving crossword puzzles from a variety of sources, such as newspapers, magazines, and contests, Dr. Fill has proven to be remarkably proficient in contrast to standard crossword-solving algorithms and is capable of functioning independently without human assistance [17].

At the heart of Dr. Fill's functionality lies a mix of computational techniques, including constraint satisfaction, pattern matching, semantic analysis, and probabilistic reasoning [17]. These techniques enable Dr. Fill to effectively decipher and interpret the often ambiguous clues presented in crosswords, strategically filling in answers with a high degree of accuracy.

This solver begins by analyzing each clue in the crossword puzzle, breaking it down into its crucial components for resolution such as keywords, word lengths, and any contextual hints provided [17]. This initial parsing step helps the solver understand the structure and requirements of each clue.

It then employs pattern matching algorithms to identify potential candidate answers that match the criteria outlined in the clue. These candidate answers are subjected to further scrutiny through constraint satisfaction techniques, ensuring they satisfy any intersecting letters and constraints imposed by neighboring answers [17].

Dr. Fill goes beyond mere pattern matching by incorporating semantic analysis techniques to assess the suitability of candidate answers in the context of the clue [17]. This involves looking at the semantic coherence and relevance of the candidate answers in comparison to the overall theme or topic suggested by the clue.

In instances in which numerous candidate answers could satisfy both its given hint and the constraints of its associated crossword grid, Dr. Fill uses probabilistic reasoning to evaluate the likelihood of each being the correct solution [17]. Factors such as word frequency, common crossword conventions, and contextual clues from neighboring answers are taken into account to assign probabilities to candidate solutions [17].

To summarize everything, its solving strategy is iterative, with the program continually refining its list of candidate answers based on feedback from intersecting clues and the overall puzzle grid [17]. As Dr. Fill progresses through the puzzle, it revisits and revises its earlier decisions, incorporating new information to improve the accuracy of its solutions.

One of Dr. Fill's key strengths lies in its extensive database, meticulously curated to encompass a vast repository of words, phrases, common crossword clues, and linguistic patterns [17]. This comprehensive knowledge base allows Dr. Fill to quickly generate potential solutions, leveraging its understanding of language semantics and crossword conventions.

Though Dr. Fill excels in executing logical and analytical tasks, its dependence on computational algorithms does raise some questions about the role of intuition, word-

play, and human ingenuity in crossword solving, a domain traditionally associated with human expertise [17].

The success of Dr. Fill in solving crosswords has garnered widespread attention and acclaim within both the crossword-solving community and the broader AI field. Its ability to tackle crosswords of varying difficulty levels, including notoriously challenging cryptic crosswords, showcases the remarkable capabilities of modern AI systems in tackling complex NLP tasks.

2.4.3 Down and Across

Another innovative method for solving crossword puzzles is offered in the paper "Down and Across: A Transformer-based Crossword Puzzle Solver," which was given at the 2022 ACL conference [19]. Driven by crosswords' appeal to the general public and the challenges they provide, this model aims to utilize advanced transformer-based language models to improve and automate crossword puzzle solving [19].

Its key innovation lies in adapting transformer architectures, known for their use in NLP, to the domain of crossword solving [19]. The model is pre-trained on a vast corpus of textual data, enabling it to learn complex linguistic patterns and semantic relationships. During the solving process, the model generates candidate solutions for each clue by leveraging its understanding of both language and context.

According to [19], their suggested method uses beam search and heuristic tactics to handle the general difficulties encountered when it comes to crossword puzzles, such as cryptic clues, wordplay, and intersecting word limitations to name a few. These strategies of theirs let the model navigate the large search space of hypothetical answers all while complying with grid limitations like word length and intersecting letters [19].

Despite its promising performance, this transformer-based crossword solver has its limits. One significant challenge it encounters is its reliance on pre-trained language models. These pre-trained language models may struggle with domain-specific nuances, as well as rare or obscure vocabulary that could take part in crosswords [19]. Addition-

ally, the computational complexity of beam search and heuristic techniques may limit the scalability of this solver when it comes to larger puzzles or real-time solving scenarios.

2.4.4 Berkeley Solver

The Berkeley Crossword Solver stands as another groundbreaking advancement in crossword-solving technology. It was developed by a team of researchers at the University of California, Berkeley and unlike previous solvers, this model integrates the most cutting-edge NLP, ML, and constraint satisfaction techniques at the time of its release to interpret and solve crossword puzzles [32]. Based on their experiments, it produced high accuracy and efficiency when it comes to crossword solving across a number of different newspapers, including the NYT [32].

One of the key distinguishing features of this solver is its incorporation of semantic analysis techniques. While earlier solvers primarily heavily relied on pattern matching and syntactic analysis, the Berkeley Crossword Solver goes beyond surface-level cues to decipher and really understand the underlying meanings that can be found within crossword clues. This enables it to select candidate answers that are not only syntactically correct but also semantically coherent and contextually relevant [32].

The Berkeley Crossword Solver also employs probabilistic reasoning algorithms to assess the likelihood of a candidate answer being the correct solution for a given clue [32]. By considering factors such as word frequency, common crossword conventions, and contextual clues from neighboring answers, the solver can assign probabilities to candidate solutions, increasing the accuracy of its predictions [32].

Additionally, the solving process of the Berkeley Crossword Solver is iterative. This method allows it to continuously refine its list of candidate answers based on feedback from intersecting clues and the overall puzzle grid [32]. This type of approach enables the solver to constantly adapt and improve its solving strategies over time, learning from past successes and failures through ML algorithms [32].

The Berkeley Crossword Solver represents a significant leap in crossword-solving technology, leveraging advanced algorithms and techniques to tackle the complexities

of crosswords with unprecedented accuracy and efficiency. Its integration of semantic analysis, probabilistic reasoning, and iterative learning sets it apart as a powerful tool for crossword enthusiasts and researchers alike, pushing the boundaries of what is achievable in automated crossword solving.

Chapter 3

Models and Solutions

In this chapter, we will be going through our general methodology, of which we will be discussing our dataset and how it came to be. Following suite, we will be describing some of the aspects that were discussed and/or employed in our model to maximize accuracy. Finally, we will be summarizing our final model's main idea.

Our general methodology consists of first collecting the necessary data, cleaning and formatting it, and finally building up our models through testing different combinations of attributes.

3.1 Datasets

To train and test our models, we utilized crosswords from the NYT that are dated from 1989 all the way up to 2018. These crosswords were part of an open source collection found on GitHub. In the following subsections, we will detail how we collected and pre-processed it.

3.1.1 Collection

Our first attempt at collecting NYT crosswords was a failure. It consisted of creating an account with the NYT and using Jack Boyce's collection tool to extract NYT crosswords [7]. However, we encountered many problems using this tool. First and foremost, it

requires a Windows or Linux based computer to be able to execute it. However, we tried many different Windows and Linux laptops and desktops to try to load the crosswords and it never worked.

As such, we turned our attention towards other sources of NYT crosswords. We came across an open source GitHub repository containing NYT crosswords from 1989 to 2018 [24]. These crosswords were formatted as JSON files and contained grid number placements, answers and clues, of which the figure below illustrates an example.

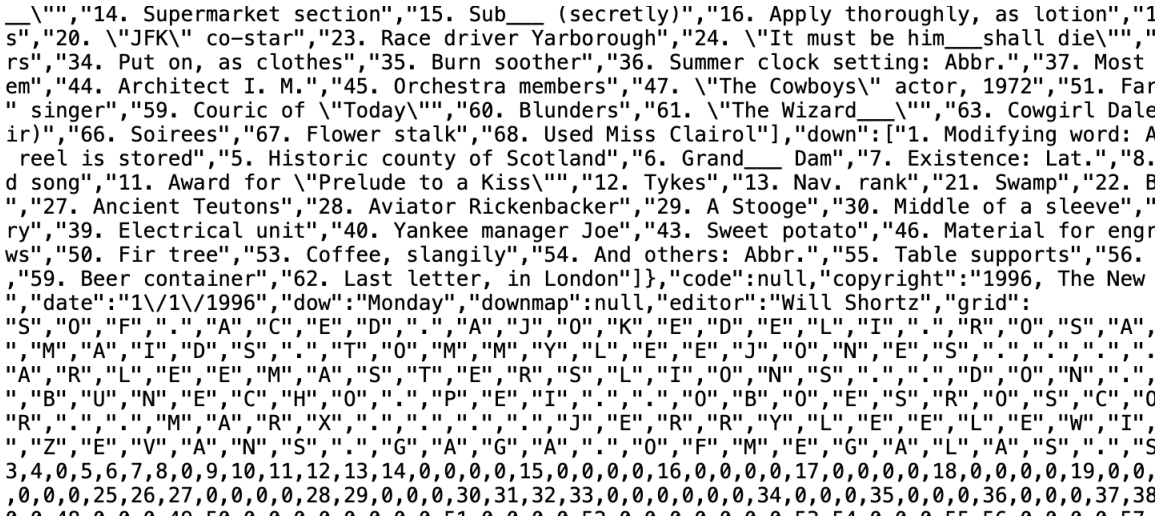

Figure 3.1: Extract from Raw Dataset

3.1.2 Pre-Processing

We pre-processed our data such that its final form is a dictionary consisting of clues and its accompanying answers look like the following:

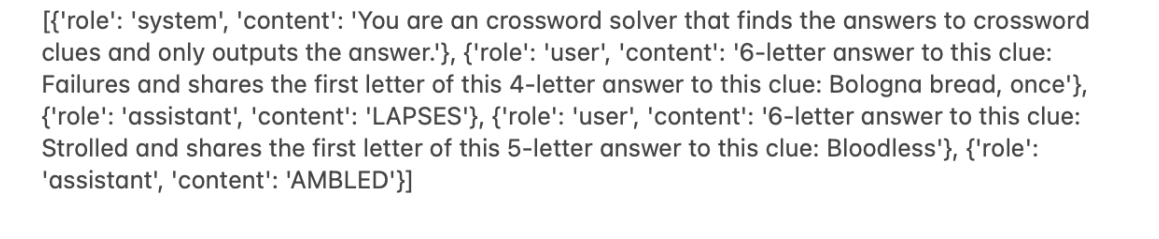

Figure 3.2: Extract from Formatted Dataset

We formatted it as such, as it is the only format that OpenAI's GPT-3.5 and GPT-4 APIs accept.

3.2 Models

When it comes to our final model, we tested a variety of different attributes that can enhance our model's performance.

We found that our best and final model consists of an amalgamation of the following: few-shot prompting, passes, grid constraints and GPT4.

Before we begin detailing our model, we would first like to preface with a few quick summaries on the following topics.

3.2.1 Few-Shot Prompting

In traditional machine learning, models are trained on large datasets with many examples of each class or category they are supposed to recognize. However, in few-shot learning, the model is trained with very few examples, sometimes just one or a handful, of each class.

Due to the large costs associated with training and fine-tuning GPT-3.5 and GPT-4 based models, we decided to go with few-shot prompting to minimize costs. To help our GPT-3.5 and GPT-4 models have a better understanding of what type of answer is expected of them and consequently reducing the need to clean our outputs, we integrated few-shot prompting into our models such that we feed them examples of both the expected input and output.

3.2.2 Majority Voting and Passes

Majority voting consists of a set of candidates and the chosen one is the one that reoccurs the most in said set. The accuracy is then based on that. Passes, on the other hand, consists of the number of times we prompt LLMs for an output.

3.2.3 Grid Contraints

Crossword grid constraints refer to the rules and limitations that govern the arrangement of words within a crossword grid. These constraints ensure that the crossword remains both challenging and solvable while maintaining consistency and coherence. Key grid constraints include:

- Intersecting Words: Words must intersect at least one letter with another word in the grid, creating both horizontal and vertical word patterns.

- Word Length: Each grid entry has a predetermined length, typically specified in the puzzle's clues. This constraint ensures that words fit within their given positions in the grid and align with possibly intersecting entries.

- Black Squares: Black squares or blocks in the grid serve as dividers between words, preventing them from intersecting where they are not intended. These squares also contribute to the puzzle's overall symmetry and iconic image.

- Symmetry: Many crossword puzzles adhere to a symmetrical layout, where horizontal and vertical entries are reflections of each other, enhancing the puzzle's visual appeal.

- Overall Grid Shape: Crossword grids often have specific dimensions, such as 15x15 or 21x21 squares for example. This has an influence on the number of entries and the overall complexity of the puzzle.

3.2.4 Final Model Algorithm

For our final model, we first reformat the crossword data. Once the data has been reformatted, we take into account the positioning of each clue in the crossword grid such that

we are aware of the length of the correct answer and any intersection of the first and last letters of the correct answer with others. We then store this information within a new dictionary, where the keys correspond to the clue's number in the grid. We also store the clues and answers in a separate dictionary with matching keys to the grid constraint dictionary.

We combine the information from both dictionaries to then feed a few entries as prompts to the LLM of our choosing so that it knows the type of output that is expected.

Once the few-shot prompting phase is over, we move onto candidate answer generation, where we let the LLM generate five candidate answers for each clue. Each of these candidate answers' word accuracy is recorded and we pick the one that is closest to the ground truth. We store all of the best candidates in a dictionary with the same keys as the previous ones.

To minimize incorrect answers, we implemented the following:

- Force the length of the candidate answer to be that of the ground truth through a feedback loop. Until an output possesses the correct length to be a candidate, we continuously prompt the LLM.

- Similarly, after all the best candidates are picked, we check if the intersecting candidates share the same first and last letters. If they do not, we continuously prompt the LLM to output another candidate until we get one with the correct length and possessing the correct intersecting letters.

Once all this is done, we calculate the final accuracy achieved for each crossword, for which we look at the number of correctly answered clues and the correct letters of the final candidate answer, as candidate answers could belong to the same word family but are ultimately not the exact same as the ground truth.

Chapter 4

Experiments and Evaluations

In this chapter, we will be detailing the testing that was done and its set-up and will conclude by showcasing the final results.

4.1 Datasets

As previously mentioned, our dataset consists of NYT crosswords taken from an open-source repository where there exists a collection of NYT crosswords dating from 1989 to 2018.

To test our model, we randomly selected 1000 of those crosswords with the help of a random date generator.

4.2 Setup

After having collected our data, we conducted a series of tests, which we describe the different models we tested and how these tests were set-up.

Before getting to our final model, we came up with the following different variations and tested all of them on a set of 1000 NYT crosswords chosen at random, where each had on average 80 clues, from an open-source repository of about 10 000 NYT crosswords:

- Baseline Model with GPT-3.5 (B3.5): This model consists of just feeding the clues to the GPT-3.5 model with the use of their API and outputting the answer.

- Baseline Model with GPT-4 (B4): Similar to the previous, the only change is that we use GPT-4 for this one.

- Few-Shot Prompting with GPT-3.5 (FS3.5): This model consists of just first feeding GPT-3.5 a few randomly selected clue-answer prompts to familiarize it with the expected input and output formats. Afterwards, we tested it by feeding the clues to the GPT-3.5 model with the use of their API and outputting the answer.

- Few-Shot Prompting with GPT-4 (FS4): Similar to the previous one, the only difference is that we employ GPT-4 for this one.

- Few-Shot Prompting with Majority Voting with GPT-3.5 (FSM3.5): Here, instead of only having one output, we generate five candidates using GPT-3.5 for each clue and pick the one that reoccurs the most often.

- Few-Shot Prompting with Majority Voting with GPT-4 (FSM4): Same as above except for the use of GPT-4 instead.

- Few Shot Prompting with 5 Passes with GPT-3.5 (FSM5P3.5): Instead of selecting the most frequently generated candidate answer, it generates five candidates using GPT-3.5 and averages out the accuracy across the five candidates.

- Few Shot Prompting with 5 Passes with GPT-4 (FSM5P4): Same as above but with GPT-4 instead.

- Few Shot Prompting with Grid Constraints with GPT-3.5 (FSG3.5): Instead of five passes, we introduce grid constraints, where we take into account the length of the answer and the intersecting first and last letters between clue-answer pairs. After few-shot prompting, we feed GPT-3.5 a clue to then generate an answer that respects the length constraint. If the generated candidate answer does not respect the length constraint, we force GPT-3.5 through a feedback loop to generate a new candidate until it does.

- Few Shot Prompting with Grid Constraints with GPT-4 (FSG4): Same as above but with GPT-4.

- Few Shot Prompting with Grid Constraints and 5 Passes with GPT-3.5 (FSG5P3.5) Similar to the previous two models, we also ask the model to generate five candidate answers that respect the length constraint and we average out the accuracy across those five.

- Few Shot Prompting with Grid Constraints and 5 Passes with GPT-4 (Final Model) Similar to above, but using GPT-4 instead.

4.3 Results

Following our set-ups, we conducted a series of tests that will be displayed further along in this thesis.

From our experiments, we have gotten the following results in Table 4.1 and Table 4.2.

Average Crossword Accuracy refers to the number of clues that were correctly answered across the 1000 crosswords. Average Word Accuracy refers to the average accuracy of each answer in terms of letters and its respective position in comparison to the ground truth.

Model Performances		
Models	Average Crossword Accuracy	Average Word Accuracy
B3.5	42.3%	46.2%
B4	47.6%	54.3%
FS3.5	46.1%	56.0%
FS4	53.7%	59.2%
FSM3.5	48.2%	57.1%
FSM4	56.5%	62.5%
FSM5P3.5	56.4%	68.3%
FSM5P4	59.3%	67.6%
FSG3.5	52.8%	68.1%
FSG4	66.7%	72.8%
FSG5P3.5	72.9%	79.2%
Final Model	83.6%	93.3%

Table 4.1: Model Variation Results

As well, we compared our final model to current well-known state of the art crossword solvers, for which the results can be found in Table 4.2.

Model Performances		
Models	Average Crossword Accuracy	Average Word Accuracy
Final Model	83.6%	93.3%
Dr. Fill	70.5%	99.4%
Berkeley	81.7%	99.7%
DownAcross	55.6%	43.4%

Table 4.2: Model Results Compared to Well-Known Solvers

4.3.1 Brief Results Discussion

From Table 4.1, among our other models, we can clearly see that our final model performed the best in both aspects. However, when it came to testing, it also took the longest to produce results.

When looking at Table 4.2 on the other hand, we see that our final model performs slightly better than the rest of them, with an Average Crossword Accuracy 1.9% higher than the Berkeley solver, which is the runner up. However, our final model is second to last when it comes to Average Word Accuracy. In terms of testing time, we are unable to compare as we were unable to run those other models due to a lack of code.

We will be discussing these results in more detail in the following chapter.

Chapter 5

Discussion

As previously mentioned, our final model did perform well. Even so, there are quite a few interesting points that we need to expand on,

Following suite from our results, in here, we will be discussing its implications, our final model's advantages and limitations in contrast

5.1 Results Discussion

In this section, we dive into two different result discussions, one for the Average Crossword Accuracy and the other for the Average Word Accuracy

5.1.1 Average Crossword Accuracy

Based on our model variations' test results, we can clearly see that our final model performed the best. In comparison to well-known solvers however, in terms of crossword accuracy, our model also performed the best out of all of them.

In general, the models using GPT-4 performed better in comparison to their counterparts using GPT-3.5. However, one interesting point to note is that FSG3.5 and FSG4 achieved worse results than FSM5P3.5 and FSM5P4 respectively.

Early on, we hypothesized that the models employing grid constraints would perform better in terms of average crossword accuracy regardless of other variations. However,

this was clearly not the case when we tested FSG3.5 and FSG4 right after FSM5P3.5 and FSM5P4, where these two models employ majority voting.

This could be due to a number of reasons. One of them is that if the GPT model misunderstood the clue, then consequently, the inclusion of grid constraints, in particular intersecting letter constraints, further dragged the GPT model away from the correct answer.

Another could be that in FSG3.5 and FSG4, we do not give it multiple passes to produce candidate answers to then choose from. Whatever the first output is, then that is the candidate answer we will use.

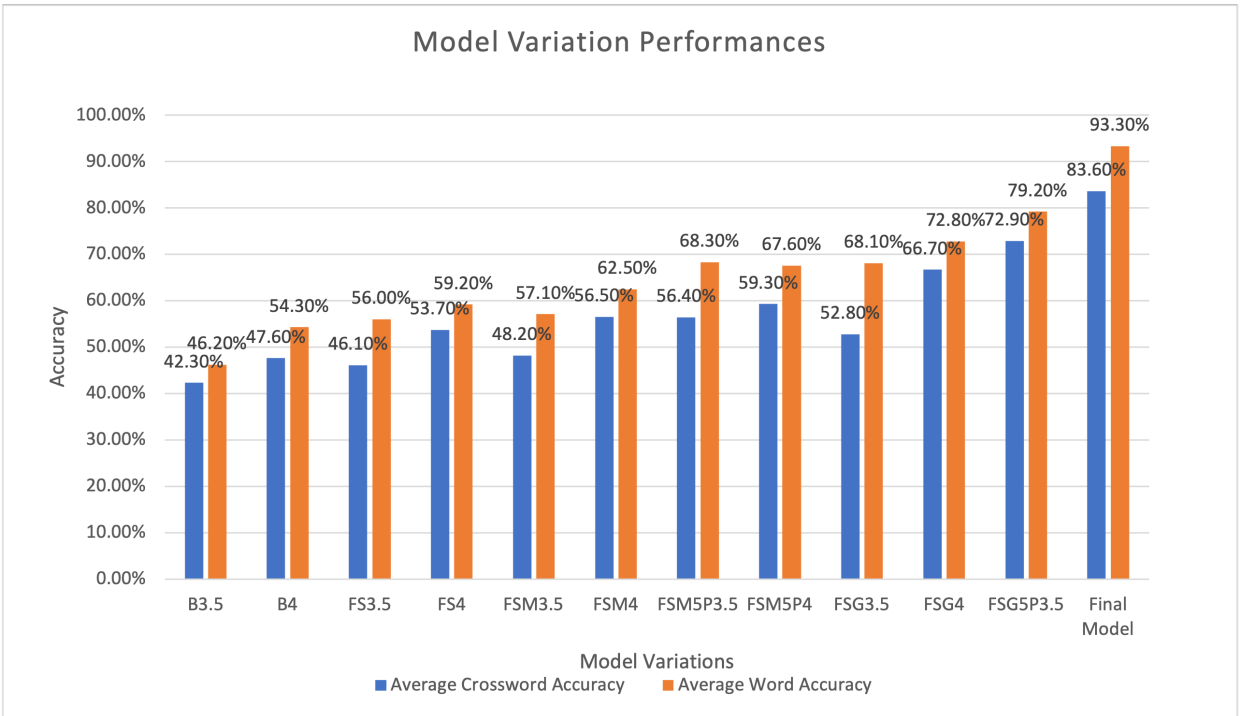

Figure 5.1: Visualization of Model Variation Performance

5.1.2 Average Word Accuracy

Moving away from the Average Crossword Accuracy, we will go through our results in terms of Average Word Accuracy in this subsection.

When it comes to the Average Word Accuracy, we find our model lagging behind Dr. Fill and Berkeley. This could be due to the fact that when our final model's predicted

answers are wrong, they are less similar to the ground truth than other solvers. It could be a consequence of including intersecting letter constraints in the inputs that we fed to our models. As previously mentioned, this can drag a candidate answer further away from the intended answer. On the other hand, previous models did not include grid constraints and mostly relied on extensive training databases to generate their answers. As such, their candidate answers are relatively similar in comparison to our final model.

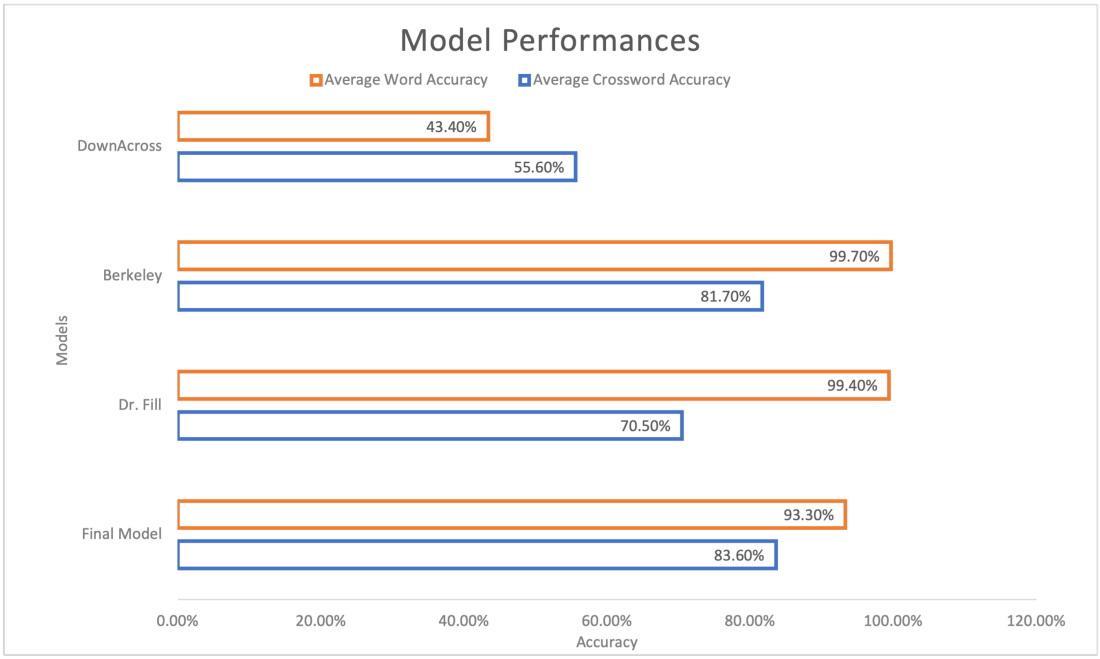

Figure 5.2: Visualization of Model Performance compared to Well-Known Solvers

5.2 Advantages

In retrospect, even though the accuracy differences between our final model and Berkeley's is not as big as we would have liked, ours does possess a key advantage over the rest of them.

Berkeley, Dr. Fill and DownAcross all require very large amounts of data for training, and consequently, this also increases the overall training time exponentially. Furthermore, this increases the amount of computational resources needed.

No training on the same scale was required for ours. As such, it makes it much less computationally heavy and time-consuming compared to theirs. With the addition of fine-tuning, we believe that our final model can achieve even higher accuracy, reach Dr. Fill's and Berkeley's and possibly beat them out when it comes to Average Word Accuracy.

5.3 Limitations

Though our model performed generally well and possessed its advantages, there are quite a few limitations that could have possibly influenced our results. The following subsections will detail two types of limitations that we considered: model and testing limitations.

5.3.1 Model Limitations

One glaring limitation is the lack of fine-tuning. All of the well-known solvers that we compared our final model to were trained and fine-tuned on hundreds of thousands to millions of rows of data. To be on a more equal footing, we would need that same amount of data to fine-tune GPT-4. However, this would cost well over tens of thousands of dollars to do so.

Another limitation is how GPT-4 and GPT-3.5 process grid constraints as prompts. Our model only takes into account the length and intersection of the first and last letters of a word. This was because we found that the more grid constraints we included in our prompts, the worse the outputs would become. As such, we lose a lot of valuable information.

Furthermore, the daily token limits imposed by OpenAI impede the amount of detail we can feed the LLMs. As such, we needed to limit both the number of words each prompt includes and the number of crosswords we tested.

5.3.2 Testing Limitations

The results of the well-known solvers were tested on a much larger number of crosswords than we have. As a result, our table is perhaps not the best for comparisons.

If there were no token or monetary limitations, we would have been able to test many more crosswords and consequently produced a more well-rounded set of results.

Even so, with the token limitations, it was not possible to test the well-known solvers on the same set of random crosswords since for two out of three of them, the code is unavailable. We tried contacting the authors but received no answers.

Furthermore, even though we performed relatively well when it comes to NYT, other solvers have also been tested on other popular journals' crosswords, such as the New York World. In spite of our success, to have a more global understanding of its performance, it would be important to see how it performs on crosswords created by different publication companies.

To add on to the importance of testing our model on additional crossword sources, the NYT has also been plagued by accusations of bias where both obscure and insensitive clues have been included. As well, a lack of diversity on the creative team has repeatedly been brought up. This can lead to a lack of diverse data to both test and train on. Altogether, it impacts the global performance of automated crossword solvers as well-rounded data is needed.

5.4 Brief Future Works Discussion

All in all, our final model possesses both advantages and limitations in comparison to other well-known crossword solvers.

Though there are limitations, the more interesting part is that there are plenty of innovations that we could incorporate into our final model in the future, such as the use of fine-tuning, using open source LLMs, dynamic programming and many more. We will be discussing these aspects more in depth in the next chapter.

Chapter 6

Conclusion and Future Works

Crosswords are a combination of art and logic. Crossword puzzle makers frequently come up with clues that lead to several viable answers that fit the grid in a coherent manner and include some challenging clues that require nuance to understand. Due to their rigorous constraints, current solvers, however, may become fixated on a particular solution and lose out on the probability of the other remaining solutions.

Certain solvers emphasize speed above all else by prioritizing computational efficiency. Although this can be advantageous, accuracy may suffer as a result, especially when the linguistic interpretation calls for a deeper comprehension. On the other hand, solvers that only focus on accuracy may take a very long time before they stop.

For our model, we combined GPT-4 along with grid constraints and few-shot prompting in order to achieve optimal results while also reducing the need for large amounts of data needed for fine-tuning, which consequently lowers the amount of time needed to produce results.

From this, we achieved our goal of outperforming the Berkeley Solver in terms of crossword accuracy. While it did not outperform its competitors when it comes to word accuracy, we did uncover some fallacies in GPT where it struggles to juggle between generating potential answers whilst also taking into account intersecting letters. We found that if more grid information is loaded into our prompts, performance rapidly worsens, such that it outputs answers.

For future research, it would be worthwhile to explore open source LLMs, how fine-tuning GPT APIs can impact accuracy, the use of dynamic programming, as well as look into cryptic crosswords, where answers are even less straightforward than traditional crosswords.

6.1 Future Works

Though our final model achieved good results, there are a few things that can be done to improve it.

In this section, we explore some aspects that can be done to improve the model, as well as looking into other types of crosswords to work on.

6.1.1 Fine-tuning

Due to their extensive pre-training on large amounts of data, LLMs are able to recognize a wide range of language patterns and information. These models may, however, need to be adjusted to fit more specialized tasks or areas. The model may learn these particular subtleties and enhance its performance on those particular jobs by fine-tuning certain datasets.

Improved accuracy when applied to tasks associated with a specific target domain is another benefit of fine-tuning a model on data from that domain in relation to a specific task. This is due to the fact that a model gains a deeper understanding of the language, terminology, and context unique to the target domain.

Furthermore, fine-tuning also allows programmers to have more control over a model's outcome. They are able to adjust hyper-parameters, such as learning rate, batch size, and number of training epochs during fine-tuning to optimize a model's performance for specific tasks or datasets.

Additionally, as more data becomes available over time, a model may be updated and enhanced continuously. Updating and adapting the model to changing language trends and patterns can be made easier through this approach.

Overall, fine-tuning serves as a powerful technique for customizing and refining LLMS in order to achieve higher accuracy and performance on a wide range of NLP tasks. Fine-tuning could enhance GPT models to adapt to specific tasks, domains, and datasets, resulting in probable significant improvements in accuracy and performance across a wide range of NLP applications.

6.1.2 Dynamic Programming

Dynamic programming is a method used to solve problems by breaking them down into simpler subproblems and solving each subproblem only once [5,9]. It stores the solutions to these subproblems in a table, allowing for efficient lookup and reuse of previously computed results [5,9].

By being able to rapidly examine and evaluate alternative solutions for specific clues, dynamic programming can help crossword solvers become more accurate. For example, when a clue is obtained, dynamic programming may be used to analyze the available letters and word patterns to generate a list of potential candidate words. Through efficient word searches and the use of previously saved answers from clue evaluations, this type of program can assist in identifying the most likely answer to a given clue. This can improve accuracy while solving crossword puzzles and assist automated crossword solvers select the optimal answer by enabling them to consider many choices.

6.1.3 Online Algorithms

Though it shares similarities with dynamic programming, another model aspect that we can take into account is online algorithms.

Online algorithms are algorithms that can analyze data as it becomes available in real-time and make judgments based on a continuous and dynamic basis without requiring knowledge of the complete input beforehand [6,21]. These algorithms are made to handle data as it comes in and make judgments continually and dynamically. In these scenarios, the algorithm must adjust to changing conditions without having access to the entire input beforehand. Online algorithms are frequently used to efficiently handle real-time pro-

cessing tasks in a variety of disciplines, such as networking, optimization, and machine learning.

By implementing these algorithms, it can help increase accuracy when it comes to crossword solvers by adapting to new information as it becomes available further down the line. These algorithms can be extremely practical in adjusting strategies in response to each new answer-clue pair that is found, making use of the letters that were revealed in real-time in the crossword grid and the restrictions imposed by each new word that intersects in order to more quickly investigate possible answers. Because of their iterative nature, predictions could be continuously improved as new data is obtained during the crossword solving process.

Overall, online algorithms provide a flexible and adaptive way to approach crossword solving, allowing the solver to make informed decisions in real-time and incorporate constraints as they become available. This dynamic strategy can help improve accuracy by efficiently exploring the solution space and iteratively refining predictions based on the constantly evolving crossword grid values.

6.1.4 Cryptic Crossword Solvers

One crossword variation that is quite popular in some parts of the world is cryptic crosswords. These types of puzzles mostly rely on linguistic tricks and wordplay. The earliest known example dates back to the early 20th century and was published in 1924 by British writer and musician Arthur Wynne. Cryptic crosswords, however, did not become widely popular until the 1930s, especially in the UK, when they started to appear more frequently in publications like The Times and The Guardian [27]. Since then, they have established themselves as a cornerstone in both print and digital media [27].

In general, they are made up of a grid of black and white squares, not unlike their traditional counterpart, where each white square corresponds to a letter in the correct word. For every word in the grid, there is a clue. These clues often include a definition or synonym of the solution, coupled with a mysterious hint on how to get there. These mysterious hints can come in a variety of shapes and sizes, such as homophones, charades,

hidden words, and anagrams, making it difficult for solvers to interpret the wordplay and discover the hidden meanings within the clues.

The solving process of cryptic crosswords involves a combination of deductive reasoning, lateral thinking, and linguistic analysis. These solvers need to carefully dissect each clue, identifying the different components of the wordplay and working backwards to derive the correct answer. This often requires a deeper understanding of language nuances, cultural references, and common crossword conventions [27].

Current solvers, while proficient in traditional clue types, may fail to adapt to the complex needs of cryptic clues [27]. There have been several automatic cryptic crossword solvers out in the market, however none have ever reached any substantial success in comparison to traditional automated crossword solvers.

Despite their intricacy, cryptic crosswords can provide a satisfying and intellectually interesting experience for puzzle fans. The satisfaction of solving a particularly difficult and purposely misleading hint is what keeps advanced puzzle enthusiasts coming back for more of these again and again. It would be worthwhile to explore this area, as it can help further train LLM to better understand natural language nuance and consequently be able to decipher hidden meanings in texts.

Copyright

Figure 2.3 was adapted from [31]. As well, we obtained permission from the NYT to use Figure 2.1.

Bibliography

[1] The new york times crossword passes 500,000 subscriptions, June 2019.

[2] ALEC RADFORD, KARTHIK NARASIMHAN, T. S., AND SUTSKEVER, I. Improving language understanding by generative pre-training.

[3] AMENDE, C. *The Crossword Obsession: The History and Lore of the World's Most Popular Pastime*. Penguin Group (USA) Incorporated, 2002.

[4] BARRETT, C., CONWAY, C. L., DETERS, M., HADAREAN, L., JOVANOVIC, D., KING, T., REYNOLDS, A., AND TINELLI, C. Cvc4. *Tool paper at CAV 2* (2011), 171–177.

[5] BELLMAN, R. *Dynamic Programming*. Princeton University Press, 1957.

[6] BORODIN, A., AND EL-YANIV, R. *Online Computation and Competitive Analysis*. Cambridge University Press, 1998.

[7] BOYCE, J. Jkboyce/nytxw$_puz : Turnnytimescrosswordsintoacrosslitefiles, 2021.$

[8] BROWN, T., MANN, B., RYDER, N., SUBBIAH, M., KAPLAN, J. D., DHARIWAL, P., NEELAKANTAN, A., SHYAM, P., SASTRY, G., AND AL. Language models are few-shot learners. In *Advances in Neural Information Processing Systems* (2020), H. Larochelle, M. Ranzato, R. Hadsell, M. Balcan, and H. Lin, Eds., vol. 33, Curran Associates, Inc., pp. 1877–1901.

[9] CORMEN, T. H., LEISERSON, C. E., RIVEST, R. L., AND STEIN, C. *Introduction to Algorithms*. MIT Press, 2009.

[10] De Moura, L., and Bjørner, N. Z3: An efficient smt solver. In *Proceedings of the Theory and Practice of Software, 14th International Conference on Tools and Algorithms for the Construction and Analysis of Systems* (2008), Springer, pp. 337–340.

[11] Dechter, R. *Constraint Processing*. Morgan Kaufmann, 2003.

[12] Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. 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)* (Minneapolis, Minnesota, June 2019), J. Burstein, C. Doran, and T. Solorio, Eds., Association for Computational Linguistics, pp. 4171–4186.

[13] Fajemisin, A. O., Maragno, D., and den Hertog, D. Optimization with constraint learning: A framework and survey. *European Journal of Operational Research 314*, 1 (2024), 1–14.

[14] Ganesh, V., and Dill, D. L. Dpll(t): Fast decision procedures. *ACM SIGPLAN Notices 44*, 10 (2009), 175–185.

[15] Gent, I. P., and Walsh, T. Easy problems are sometimes hard. *Artificial Intelligence 70*, 1 (1994), 335–345.

[16] Gent, I. P., and Walsh, T. The sat problem: An overview. In *Handbook of satisfiability*. IOS Press, 2006, pp. 19–153.

[17] Ginsberg, M. L. Dr.fill: Crosswords and an implemented solver for singly weighted csps. *ArXiv abs/1401.4597* (2011).

[18] Hooker, J. *Integrated Methods for Optimization*, vol. 170. 01 2012.

[19] Kulshreshtha, S., Kovaleva, O., Shivagunde, N., and Rumshisky, A. Down and across: Introducing crossword-solving as a new NLP benchmark. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*

(Dublin, Ireland, May 2022), S. Muresan, P. Nakov, and A. Villavicencio, Eds., Association for Computational Linguistics, pp. 2648–2659.

[20] LITTMAN, M. L., KEIM, G. A., AND SHAZEER, N. M. Solving crosswords with proverb. AAAI '99/IAAI '99, American Association for Artificial Intelligence, p. 914–915.

[21] MANASSE, M., AND MCGEOCH, L. A. Smoothed analysis of algorithms: Why the simplex algorithm usually takes polynomial time. *Proceedings of the 28th ACM Symposium on Theory of Computing* (1994).

[22] MOSKEWICZ, M. W., MADIGAN, C. F., ZHAO, Y., ZHANG, L., AND MALIK, S. Chaff: Engineering an efficient sat solver. *Proceedings of the 38th Design Automation Conference* (2001), 530–535.

[23] OPENAI, :, ACHIAM, J., ADLER, S., AGARWAL, S., AHMAD, L., AKKAYA, I., ALEMAN, F. L., AND AL. Gpt-4 technical report, 2023.

[24] O'SHEA, D. Doshea/nyt$_c rosswords : Everynytcrosswordinjsonformat, 2018.$

[25] RADFORD, A., WU, J., CHILD, R., LUAN, D., AMODEI, D., AND SUTSKEVER, I. Language Models are Unsupervised Multitask Learners.

[26] RAFFEL, C., SHAZEER, N., ROBERTS, A., LEE, K., NARANG, S., MATENA, M., ZHOU, Y., LI, W., AND LIU, P. J. Exploring the limits of transfer learning with a unified text-to-text transformer.

[27] RICHARDSON, A. The penguin book of canadian cryptic crosswords. *Canadian Book Review Annual Online*. accessed February 10, 2024,.

[28] SAGAL, P. Here's looking at you, grid: A history of crosswords and their fans, Mar 2020.

[29] SLANEY, J., AND WOLTZENLOGEL PALEO, B. Conflict resolution: a first-order resolution calculus with decision literals and conflict-driven clause learning. *Journal of Automated Reasoning 60* (2018), 133–156.

[30] TSANG, E. *Foundations of Constraint Satisfaction*. Academic Press, 1993.

[31] VASWANI, A., SHAZEER, N., PARMAR, N., USZKOREIT, J., JONES, L., GOMEZ, A. N., KAISER, L. U., AND POLOSUKHIN, I. Attention is all you need. In *Advances in Neural Information Processing Systems* (2017), I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, Eds., vol. 30, Curran Associates, Inc.

[32] WALLACE, E., TOMLIN, N., XU, A., YANG, K., PATHAK, E., GINSBERG, M., AND KLEIN, D. Automated crossword solving. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)* (Dublin, Ireland, May 2022), S. Muresan, P. Nakov, and A. Villavicencio, Eds., Association for Computational Linguistics, pp. 3073–3085.

[33] YANG, Z., DAI, Z., YANG, Y., CARBONELL, J., SALAKHUTDINOV, R., AND LE, Q. V. *XLNet: generalized autoregressive pretraining for language understanding*. Curran Associates Inc., Red Hook, NY, USA, 2019.