Creating a Crossword Puzzle Solver Using Natural Language Processing

Jonathan L. Merrill

December 8, 2022

Abstract

Background: For over 100 years now, people have been trying to do whatever it takes to find the answers to crossword puzzles. The pinnacle of all crossword puzzle solvers is the Berkley Crossword solver, which was able to outperform 1100 of the top crossword solvers in the world.
Goals: The overall goal of this project was to produce a model that can generate crossword answers based on given clues that is accurate, fast, and does not depend on previous crossword data.
Methods: The first task was trying to understand how a computer might understand and come up with answers based on crossword clues. To make the task easier, the clues were split up into five types, each of which has its own method of solution.
Results: The overall goal of the project was achieved with about 72% accuracy. The second goal, to not use any prior data, failed. The last goal, complexity, was also achieved. The model was trained with large layers, many LSTM units, a large batch size, and lots of epochs. Once the model is trained, the results can be achieved almost instantly.
Conclusions: The Neural Network that was implemented was good. There could be major improvements and additional methods tested. The culmination of this research would be to combine both the Neural Network and the data-less strategy to work with each other to create the most effective crossword puzzle solver.

Background and History

History of the Crossword Puzzle

In December of 1913, a section editor for the New York World newspaper named Arthur Wynne was tasked with filling a Christmas insert for the newspaper. He drew back to his English roots and decided to give his spin on a game published in Victorian newspapers called word squares, in which a puzzle has acrostic answers where words can be read both across and down (Mansky, 2022). Readers enjoyed the challenge of Wynne's so-called 'crossword puzzle' and began writing in suggestions for future puzzles and complaints about past puzzles. The crossword was never meant to be a mainstay of the newspaper, but the people quickly became too attached to easily let it go, and the newspaper was flooded with hundreds of letters when it failed to appear one Sunday. Wynne resented the task and eventually passed it off to others in the newspaper.

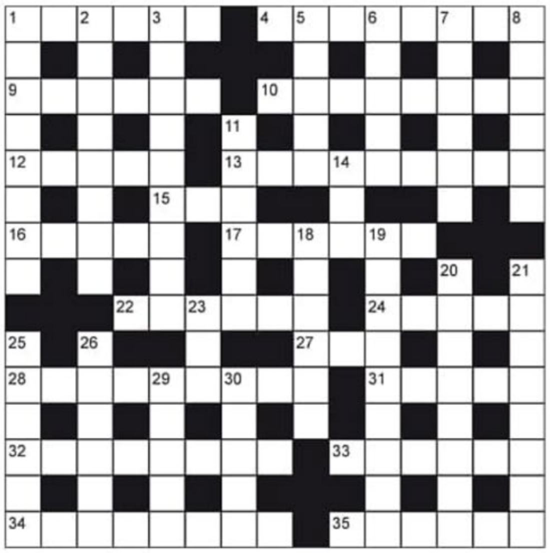
In the early 1920s, crossword puzzles began to explode in popularity. "Crosswords were the Beatles of 1924," the second editor of the original crossword later said (Mansky, 2022). Other newspapers began to publish their own version of the crossword, and a crossword book was even published with remarkable success. They became popular enough that families began to use them to announce engagements, and other news stories were announced in the crossword. Much as wordle took the world by storm in early 2022, crossword puzzles were the talk of the public in the early 1920s, even inspiring this comic in 1925 (Mansky, 2022):

THE CROSS-WORD MANIA.

Doctor (rung up at 2 a.m.) "Yes, Dr. Brown speaking. What is it?"
Voice. "I want the name of a bodily disorder of seven letters, of which the second letter must be 'N.'"

Beyond just being a fun pass-time, the crossword has come into historical importance a few times. In 1942, after having refused to publish their own version of the crossword for over 20 years, the New York Times finally published its first crossword puzzle to "provide readers something to occupy time during the coming blackout days." (Raphael, 2020). The crossword was seen as a mental distraction from the stresses of the war. On the other side of the Atlantic, some British people were complaining that their crossword puzzles were becoming too easy and implored a newspaper to make the puzzles more difficult. In response, the newspaper invited anyone to come to try and conquer a new crossword puzzle in under 12 minutes. Of the many people that came, only five finished within the given time. The next day, the newspaper published the crossword puzzle and challenged the general public to see if they could complete the challenge. A few weeks after the event, the fastest solvers received a letter from Bletchley Park (the code-breaking division of the British military during World War 2) offering them work. That crossword puzzle is seen below (Adams n.d.):

TELEGRAPH CROSSWORD 5,062
13 JANUARY 1942

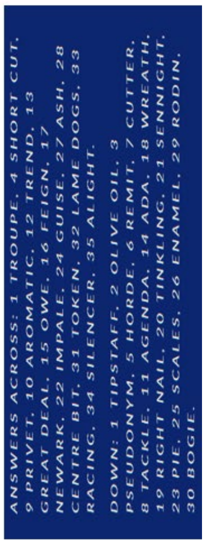


Across

1 A stage company (6)
4 The direct route preferred by the Roundheads (5,3)
9 One of the ever-greens (6)
10 Scented (8)
12 Course with an apt finish (5)
13 Much that could be got from a timber merchant (5,4)
15 We have nothing and are in debt (3)
16 Pretend (5)
17 Is this town ready for a flood? (6)
22 The little fellow has some beer; it makes me lose colour, I say (6)
24 Fashion of a famous French family (5)
27 Tree (3)
28 One might of course use this tool to core an apple (6,3)
31 Once used for unofficial currency (5)
32 Those well brought up help these over stiles (4,4)
33 A sport in a hurry (6)
34 Is the workshop that turns out this part of a motor a hush-hush affair? (8)
35 An illumination functioning (6)

Down

1 Official instruction not to forget the servants (8)
2 Said to be a remedy for a burn (5,3)
3 Kind of alias (9)
5 A disagreeable company (5)
6 Debtors may have to this money for their debts unless of course their creditors do it to the debts (5)
7 Boat that should be able to suit anyone (6)
8 Gear (6)
11 Business with the end in sight (6)
14 The right sort of woman to start a dame school (3)
18 "The war" (anag.) (6)
19 When hammering take care not to hit this (5,4)
20 Making sound as a bell (8)
21 Half a fortnight of old (8)
23 Bird, dish or coin (3)
25 This sign of the Zodiac has no connection with the Fishes (6)
26 A preservative of teeth (6)
29 Famous sculptor (5)
30 This part of the locomotive engine would sound familiar to the golfer (5)



ANSWERS ACROSS: 1 TROUPE, 4 SHORT CUT, 9 PRIVET, 10 AROMATIC, 12 TREND, 13 GREAT DEAL, 15 OWE, 16 FEIGN, 17 NEWARK, 22 IMPALE, 24 GUISE, 27 ASH, 28 CENTRE BIT, 31 TOKEN, 32 LAME DOGS, 33 RACING, 34 SILENCER, 35 ALIGHT.

DOWN: 1 TIPSTAFF, 2 OLIVE OIL, 3 PSEUDONYM, 5 HORDE, 6 REMIT, 7 CUTTER, 8 TACKLE, 11 AGENDA, 14 ADA, 18 WREATH, 19 RIGHT NAIL, 20 TINKLING, 21 SENNIGHT, 23 PIE, 25 SCALES, 26 ENAMEL, 29 RODIN, 30 BOGIE.

Also, during World War 2, some of the answers to crossword clues worried British officers because they were the top secret code names for aspects of battle plans. In May of 1944, the words “UTAH”, “OMAHA” (2 of the code names for beaches of the coming D-Day attacks), “MULBERRY” (the operation’s floating harbors), “NEPTUNE” (the naval assault stage), and “OVERLORD” (the codename for D-Day itself) were all used as answers to the crossword. All of these answers were traced back to one crossword writer who the British officials were sure was a spy. It was discovered 40 years later that this man (who was also a school teacher), was getting clue ideas from his students who had been spending time around British soldiers and had given their teacher clue ideas based on some of the obscure words they had heard them saying. All in all, a few crossword puzzles almost gave away top-secret information at a very pivotal time that could have changed the tide of the war (Raphael, 2020).

Prior Crossword Solving Using Natural Language Processing

For over 100 years now, people have been trying to do whatever it takes to find the answers to crossword puzzles. With the invention of computers and higher-powered computing, this quickly became a task for early programmers. With the advent of the internet, where people must have the answers to their every question at a moment's notice, and the digitization of newspapers, crossword puzzle solvers became more prevalent. A simple google search will get you hundreds of options for finding that tricky crossword answer in an instant. As far as I can tell, most, if not all, of these methods are simple look-up methods that return the most common results with any given clue from a bank of all previous crossword puzzle clues. These methods might not be the most collectively exhaustive, but they’ll get the job done for 99% of amateur needs.

The step above these simple internet methods is a collection of research done with the same idea, but with much more complex methods. Researchers get a bank of millions of crossword answers and use machine learning methods to find the most likely answers. These range from very basic machine learning algorithms to very complex methods that involve high-level mathematics and complex programming. Some of the better methods involve Monte Carlo Tree Search, or Hidden Markov Models employed on top of a Neural Network to optimize the results (Chen et al., 2022).

The pinnacle of all crossword puzzle solvers is the Berkley Crossword solver (Wallace et al., 2022). At its core, finding the answers to crossword puzzles is a natural language processing problem. Many others have used elements of Natural Language Processing in their solutions, but most of their work is more about Machine Learning and less about Natural Language Processing. The team at Berkley created their solution more heavily based on Natural Language Processing (Wallace et al., 2022). While they do use a complex machine learning model trained on previous crossword data, they also use a very complex natural language AI called GPT-2 (the predecessor of GPT-3, which is currently taking the world by storm with ChatGPT), on Wikipedia articles to collect question and answer pairs which they also use in their models. The Berkley Crossword solver is incredibly impressive and was able to outperform 1100 of the top crossword solvers in the world. This research has become the benchmark for all future crossword puzzle research (Wallace et al., 2022).

Goals

In the quest to create an effective crossword puzzle solver, this project has three main goals. The first goal of this project is accuracy. Accuracy is the clear measure of the effectiveness of a method. No matter how much work is put in, and no matter how elegant the concepts, it really means nothing if the model is not able to perform with good accuracy. Beyond this, the goal is that the method produced will have broad accuracy and not just good accuracy on specific subsets of tests.

The second goal of this project is to come up with a method that does not rely on previous crossword data. Seeing as this is a Natural Language Processing project, I wanted the main focus of the project to be about Natural Language Processing and less about being a machine learning project that happens to have English words in it like so many have done before.

The last goal of this project is to minimize the temporal complexity. Beyond accuracy, the second most important aspect of the effectiveness of a model is the speed at which it computes answers. It really doesn't matter if your model is effective if it takes an hour to produce a single result. All these combined, the overall goal of this project is to produce a model that can generate crossword answers based on given clues that is accurate, fast, and does not depend on previous crossword data. Accurate and fast are fairly relative terms, but seeing as this is uncharted territory, I am okay leaving those terms as relative.

Methodology

The first task was trying to understand how a computer might understand and come up with answers based on crossword clues. Crossword puzzles are intentionally made tricky, so this became quite a difficult task. To make the task easier, the clues were split up into five types, each of which has its own solution to solve.

The first type of clue is 'for example' clues where the reader is looking for an answer that is an example of the item given in the clue. An illustration of this is the clue "Rainbow or brook, e.g.," could have the answer TROUT. Any clues that contained the words "For example" or "e.g." were grouped into this category. To find potential answers to these types of clues, I used NLTK in python to generate wordnet synset example sentences and gathered all the words that matched the desired length to a bank of words. This process was repeated for all the words in the clue, and the model then predicted the correct answer based on which word with the matching length was used the most frequently.

The second type of clue is 'fill-in-the-blank' clues. This is any clue that contains an underscore character in which the reader is trying to predict the word that is missing. An example of this is the clue "___ of a kind," which could have the answer ONE. To do this, I used n-grams of varying lengths trained on articles published by the New York Times (which seemed fitting because the crossword puzzles were published by the New York Times). N-grams is a tool that groups phrases of length 'n' together and can then predict which words will come next based on the frequency in which those words are grouped together. This is essentially what is being done in 'fill-in-the-blank' clues, so this seemed like the obvious method to use. Additionally, if the blank was in the middle of the sentence, a double validation method was used. First, the word was predicted with the letters before the blank, then the letters after the blank, and finally using the whole sentence. Whichever answer was the most common after all three tests was the predicted answer.

The third type of clue is 'Proper Noun' clues. These are clues that contain a proper noun. An example of this might be the clue "Famous Golfer," which could have an answer WOODS. It became necessary to split these clues into their own category after it was found that the model was having trouble trying to find synonyms and definitions of proper noun words, which made retrieving answers very difficult. These clues were treated very similarly to the 'for example' clues, where WordNet example sentences were used to create a bank of words of the correct length, and then predictions were made based on frequency.

The fourth type of clue is 'Anagram' clues. This is a clue in which the answer is a rearrangement of the letters. An example of this might be the clue "An anagram for 'staple'" could be PASTEL. These clues were handled by partitioning every combination of the letters of the word and retaining those that were valid words.

All other clues were handled by using WordNet to find the definitions and example sentences of every hypernym, hyponym, synonym, and antonym of every word in the clue and adding the words with the matching length to a bank of words, and then predicting the answer based on the number of occurrences of each word in the bank.

Results

The results of each of these methods varied greatly. In the beginning, the ‘for example’, ‘proper noun’, and ‘other’ clues were struggling to get the correct word into the bank of words from which predictions were being made. To combat this, I added the n-gram method to help add more words to the bank. This improved the ability of the bank of words to contain the correct words for ‘proper noun’ and ‘for example’ clues, but did not work well for the ‘other’ clues. This makes sense because the answer to ‘proper noun’ clues and ‘for example’ clues are words that are much more likely to occur in the same sentence as the clue, whereas many of the clues from the ‘other’ category are associated words that would be less likely to be used in the same sentence.

After the adjustments, the method for the ‘for example’ clues was able to get the correct answer into the bank of words about 70% of the time and was able to accurately predict the answer about 40% of the time. In addition, ‘proper noun’ clues were able to get the correct word into the bank about 65% of the time and accurately predicted the correct answer about 50% of the time. The method for ‘fill-in-the-blank’ clues was much more effective, being able to predict the correct answer about 85% of the time. Further, the method for ‘anagram’ clues was able to predict the correct answer about 98% of the time. Lastly, the method for all the ‘other’ clues was only able to get the correct answer in the bank of words about 30% of the time, and when the answer was in the bank, it was only predicting that answer about 45% of the time for a total of about 13% success.

I was satisfied with the results from the first four types of clues and focused my main efforts on trying to figure out more effective ways to predict the answer for the ‘other’ clues category. The main issue with the ‘other’ clues was that 70% of the time, the bank of words did not contain the correct answer. Because the correct word was only in the bank 30% of the time, the best accuracy the model could achieve was 30%. To increase the bank of words, I added a level of recursion to my method and began finding the definitions, example sentences, hypernyms, hyponyms, synonyms, and antonyms of every word in the complete bank of words (not just the ones matching the length). I then added the words from the new list that matched the correct length to a new bank of words. This was much more effective at getting the correct word into the bank, but at quite a cost. Before this implementation, the bank of words was usually about 40-50 words long at most, but after this implementation, the bank of words was up to 1000 words long for some of the longer clues. So, while the ability to get the word in the bank had increased, the accuracy of predicting that word from the bank had plummeted to near 0, and the method now took 10x longer to run.

Improvements

To improve the method for the ‘other’ clues, I then tried to work backward and put the answers in as clues to see if the model could predict any of the words from the clues. I did this to test if my model had any credibility at even associating clues and answers. The model was much more successful with this, achieving at least one of the words from the clue about 60-70% of the time. This was quite discouraging for two reasons. First, the model was only able to connect the words from the answer and the question about 65% of the time, even when being spoon-fed the inputs. This inherently meant there was something wrong with the method and not just the implementation. Using all the closest related words that WordNet could produce was simply not effective enough at finding the attached words that crossword was looking for. Second, it is virtually impossible to implement this process going forward because the backward method would need to be implemented on every word in the word bank (which already only contained the right word 30% of the time). Then, even with that, there would be other words that were connected closer, causing the model to predict the wrong word. Not to mention, the time complexity of implementing this method would increase the time it takes to run the program by about 100x.

Complications

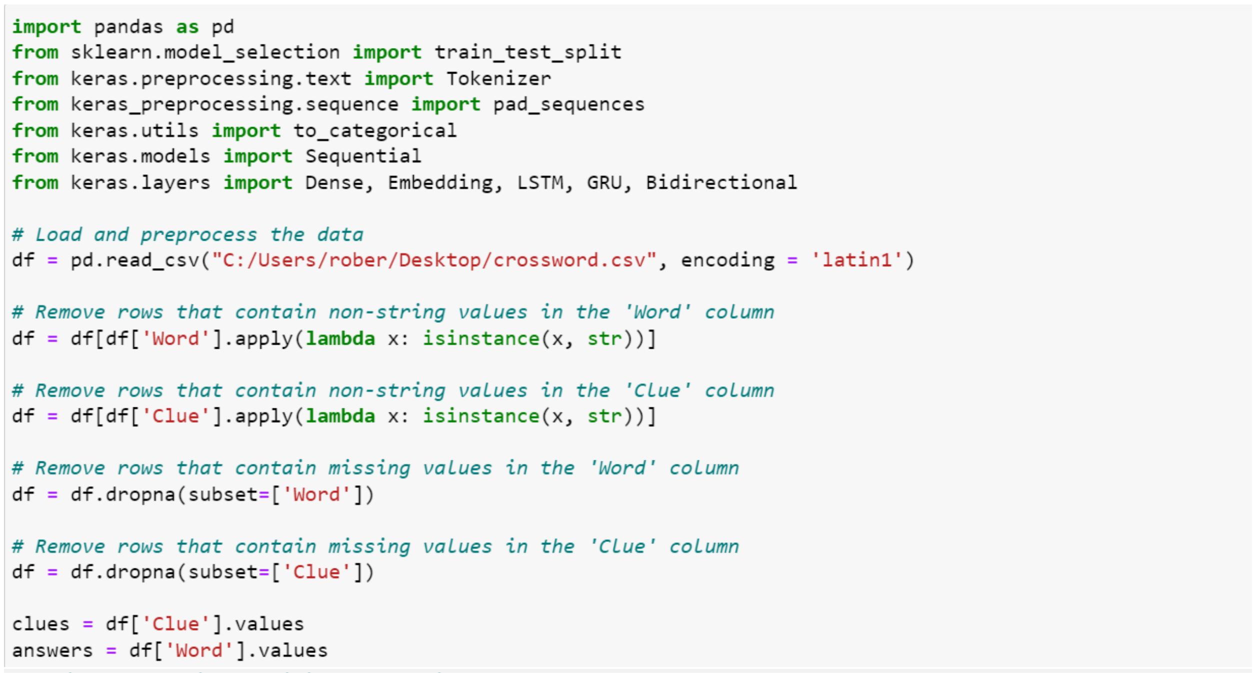
As a result of these conclusions, it became clear that to find success, a new method must be sought. To find a better model, I must increase my understanding of crossword puzzles and how the connections are made between the clues and answers. After extensive research and much failed implementation, I found that there were about 9 major obstacles in crossword puzzle clues that made guessing the answers nearly impossible. These are:

1. Cryptic Clues (clue "cost of sitting" = STANDING CHARGE)
2. Combined word answers (clue "pickle" = INA (as in "in a pickle"))
3. Sound Clues (clue "The king has no successor, we hear" = BALD (because no heir sounds like no hair))
4. Hidden Words (clue "Animal or Beaker in warm-up petticoats = MUPPET (war**m-up pet**ticoats)
5. Unprompted anagrams (often uses words like scrambled/tossed/altered/rearranged/confused/bizarre/strange/renovated etc.) (clue "Christmas shrub brewed in its teapot" = POINTSETTA which is an anagram of "in its teapot")
6. Roman Numerals (clue "I, II, III etc" = ROMAN NUMERALS)
7. Abbreviations (N, E, S, W abbreviating directions or in music p abbreviating quietly and f loudly)
8. 'embraces', 'surrounds', 'in' (words like this indicate hidden words or other tricky rearrangements like clue "The Parliamentarian surrounds a duck" = MOP or clue "a long time in a dungeon" = EON)
9. Title abbreviations ('Navy' = RN, 'Engineers' = RE, 'Conservative' = C, 'Doctor' = DR or GP, 'Resistance' = OHMS etc.)

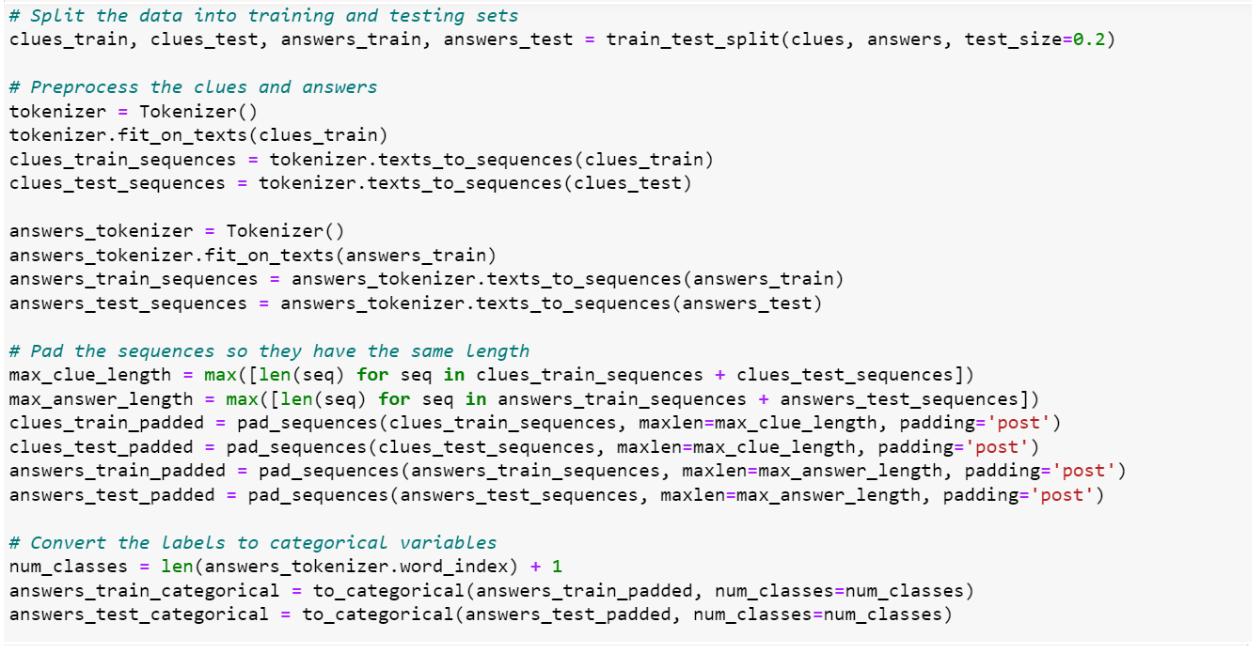
Despite these obstacles, being the stubborn person that I am, I did not give up and banged my head against this problem for a few weeks trying everything I could think of to no avail. It was at this point that I realized the only way to get any decent results out of this was to sacrifice the goal of using previous crossword data and create a neural network that predicted results based on previous crossword answers.

Neural Network

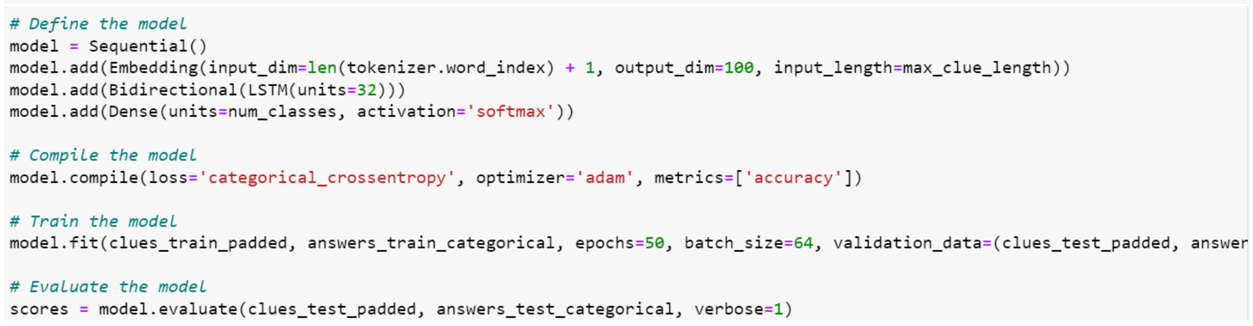
The implementation of the neural network was nothing fancy; in fact, it was very simple and largely taken from the ideas of the code from the Berkeley project. Since my CPU pales in comparison to that of the expensive computers owned by Berkely, I had to make the code rather simple, and this is what I ended up with:



```python
import pandas as pd
from sklearn.model_selection import train_test_split
from keras.preprocessing.text import Tokenizer
from keras_preprocessing.sequence import pad_sequences
from keras.utils import to_categorical
from keras.models import Sequential
from keras.layers import Dense, Embedding, LSTM, GRU, Bidirectional

# Load and preprocess the data
df = pd.read_csv("C:/Users/rober/Desktop/crossword.csv", encoding = 'latin1')

# Remove rows that contain non-string values in the 'Word' column
df = df[df['Word'].apply(lambda x: isinstance(x, str))]

# Remove rows that contain non-string values in the 'Clue' column
df = df[df['Clue'].apply(lambda x: isinstance(x, str))]

# Remove rows that contain missing values in the 'Word' column
df = df.dropna(subset=['Word'])

# Remove rows that contain missing values in the 'Clue' column
df = df.dropna(subset=['Clue'])

clues = df['Clue'].values
answers = df['Word'].values
```



```python
# Split the data into training and testing sets
clues_train, clues_test, answers_train, answers_test = train_test_split(clues, answers, test_size=0.2)

# Preprocess the clues and answers
tokenizer = Tokenizer()
tokenizer.fit_on_texts(clues_train)
clues_train_sequences = tokenizer.texts_to_sequences(clues_train)
clues_test_sequences = tokenizer.texts_to_sequences(clues_test)

answers_tokenizer = Tokenizer()
answers_tokenizer.fit_on_texts(answers_train)
answers_train_sequences = answers_tokenizer.texts_to_sequences(answers_train)
answers_test_sequences = answers_tokenizer.texts_to_sequences(answers_test)

# Pad the sequences so they have the same length
max_clue_length = max([len(seq) for seq in clues_train_sequences + clues_test_sequences])
max_answer_length = max([len(seq) for seq in answers_train_sequences + answers_test_sequences])
clues_train_padded = pad_sequences(clues_train_sequences, maxlen=max_clue_length, padding='post')
clues_test_padded = pad_sequences(clues_test_sequences, maxlen=max_clue_length, padding='post')
answers_train_padded = pad_sequences(answers_train_sequences, maxlen=max_answer_length, padding='post')
answers_test_padded = pad_sequences(answers_test_sequences, maxlen=max_answer_length, padding='post')

# Convert the labels to categorical variables
num_classes = len(answers_tokenizer.word_index) + 1
answers_train_categorical = to_categorical(answers_train_padded, num_classes=num_classes)
answers_test_categorical = to_categorical(answers_test_padded, num_classes=num_classes)
```



```python
# Define the model
model = Sequential()
model.add(Embedding(input_dim=len(tokenizer.word_index) + 1, output_dim=100, input_length=max_clue_length))
model.add(Bidirectional(LSTM(units=32)))
model.add(Dense(units=num_classes, activation='softmax'))

# Compile the model
model.compile(loss='categorical_crossentropy', optimizer='adam', metrics=['accuracy'])

# Train the model
model.fit(clues_train_padded, answers_train_categorical, epochs=50, batch_size=64, validation_data=(clues_test_padded, answer

# Evaluate the model
scores = model.evaluate(clues_test_padded, answers_test_categorical, verbose=1)
```

Analysis

To evaluate the success of this project, it is important to review the initial three goals. The first goal of accuracy was achieved. I was able to reach about 72% accuracy. Obviously, I would like to be more accurate and make a better model, but considering the time that was allocated to the neural network, the project's accuracy should be considered a success. The second goal was not to use any prior data. This goal failed. Sadly, this aspect of the project was unable to be completed in the months allotted. Perhaps with more time, this could be possible, but it seems that the goal was too lofty for a semester project done by one person. The last goal was complexity. This goal was also achieved. For this model to be successful, it was trained with large layers, many LSTM units, a large batch size, and lots of epochs. This makes the model take a long time to train, but once it has been trained, the results can be achieved almost instantly. For this reason, I would consider this a success in this goal.

Future Work and Conclusion

There is much that can be done in the future to improve this project. I am convinced that there is a way that this can be done without using prior data from crossword puzzles. However, the more research I do, the more I realize how complex of a Natural Language Processing problem this is. But, with the advent of GPT-3, and the field growing, it is likely that resources will only continue to grow in the field of Natural Language Processing, and more tools will become easily accessible to college students. In terms of the Neural Network, there could obviously be many improvements. The Neural Network that was implemented was good considering the time spent, but given more time, there could be major improvements and a lot of additional methods tested. The culmination of this research would be to combine both the Neural Network and the data-less strategy to work with each other to create the most effective crossword puzzle solver that could eventually outperform humans 100% of the time.

References

Chen, L., et al (2022). Crossword Puzzle Resolution Monte Carlo Tree Search. *Proceedings of the International Conference on Automated Planning and Scheduling*. Vol. 32. https://ojs.aaai.org/index.php/ICAPS/article/view/19783

Mansky, J. (2022). A Century Before Worlde Went Viral, Crossword Mania Swept the Country. *Smithsonian Magazine*. https://www.smithsonianmag.com/arts-culture/a-century-before-wordle-went-viral-crossword-mania-swept-the-country-180979858/

Raphael, A. (2020). Crosswords Have Always Been a Solace in Times of Trouble. Here's How the 20th Century's Toughest Moments Shaped the Puzzle's History. *Time Magazine*. https://time.com/5811396/crossword-history/

Stephen, A. (n.d.) The Cryptic Crossword that Recruited for Bletchley Park. http://www.alaricstephen.com/main-featured/2016/9/18/the-cryptic-crossword-that-recruited-for-bletchley-park

Wallace, E. et al. (2022). Automated Crossword Solving. *arXiv*. https://doi.org/10.48550/arXiv.2205.09665