Project 1 - CNIT 51900 Fall 2023

Group: 1

Team Members: Melissa Kronenberger, Vinicius Lima, Aidan Sommers, Adrienne Zhang

Professor: Julia Rayz

Pun Detection and Interpretation

1. Introduction

1.1. Project Objectives

The task was to write a program that could detect sentences in which words or phrases have multiple interpretations. The project was intentionally open-ended, and designed to encourage learners to play with word senses and meaning. Puns were indicated to be excellent examples of sentences bearing this type of double-meaning, and the instructions welcomed us to focus on puns. The scope was limited to homographs, which are words that share the same written form, rather than homophones, which are words that differ in form but may still share the same or similar pronunciation.

For the purposes of this project's scope, detecting a pun meant to identify which word in a given sentence, if any, was being used with two different meanings, and to reasonably indicate how these meanings were in use. If no such word was detected, the program was to declare such. For example, in the sentence, "I've been to the dentist many times, so I know the drill," the program must identify 'drill' as the pun word, with its corresponding senses being 'a rotary tool used in dentistry' and 'a pattern of actions.'

1.2. Resources

For the purpose of training and testing machine learning algorithms, large language models, and any other tools we elected to utilize over the course of this project, we found it prudent to seek out one or more datasets. Our ideal dataset would contain puns and non-puns, as well as information about pun detection, pun location, and pun interpretation. Ideally, pun interpretation would be expressed in a standardized format, which our program would then be able to train on and subsequently be evaluated on.

To this end, the most valuable resource we encountered was *SemEval-2017's Task 7: Detection and Interpretation of English Puns* (Miller et al., 2017). This was a shared task put forward by the International Workshop on Semantic Evaluation, and we located it after following the citation trail of a large number of related GitHub repositories and published papers. Not only did Task 7's participants each publish a paper on their results, but the workshop itself published an overview paper to summarize them. The task had prepared resources for its participants, including a dataset of homographic puns and the detection, location, and interpretation of those puns. Interpretations were given in WordNet word senses, providing us with a standardized mechanism for training models.

Over the course of the project, we assessed that our models were better at interpreting puns than at detecting whether a pun existed at all. Our program frequently misclassified proverbs, idioms, and other common sayings as puns. As we found it necessary to place emphasis on pun detection, we tapped two additional resources. The first was a database of puns scraped from a Reddit thread on humor, in which user-submitted jokes were rated by other users (Weller & Sepppi, 2019). The second was a database of proverbs, which we hoped would help train our algorithms to distinguish between puns and phrases that merely resembled puns (Lückert, 2019).

1.3. Program Architecture

To achieve the project objectives, we brainstormed various program architectures and employed an iterative approach to testing several different methods for pun classification and interpretation. In doing so, we learned about the structure of puns, investigated word associations, and explored a wide range of tools and concepts.

First, in an attempt to apply concepts from the course, we looked to use WordNet to compute the similarity between various words within the pun. This approach gave us substantial technical experience in utilizing WordNet and the Natural Language Toolkit, but proved insufficient for associating words with double meanings relative to the different contexts. Even so, dabbling with WordNet proved valuable, as we continued to take advantage of word senses when training on and testing against the dataset provided by SemEval-2017's Task 7.

Next, we looked to test several different methods of pun detection and interpretation that utilize large language models (LLMs). These models gave us the ability to work with reasoning and association relative to each pun's context. To test the differences across approaches, we worked with large language models by:

1. Fine-tuning a relatively smaller model on our dataset of puns
2. Prompting a model specifically for yes/no responses for pun identification
3. Prompting a model with pun examples and unformatted explanations
4. Prompting a model with formatted step-by-step explanations for how to break down puns

Finally, we evaluated qualitative strengths and weaknesses of our work with various LLM prompts and fine-tuning. After doing so, we decided to temporarily narrow our scope to pun classification, in an effort to mitigate our program's weaknesses. Under this reduced scope, we trained two machine learning algorithms to distinguish between puns and non-puns. Each algorithm exhibited slightly better performance than any of the LLMs. Our final program is a script that identifies if a given context includes a pun, breaks down puns into contextual components, highlights a word with multiple interpretations across the given context, and explains the double meaning of the relevant word.

2. Methods

2.1. Investigating Puns through WordNet Similarity

Our initial approach to addressing the task involved exploring the various senses that a word can possess, as defined by WordNet. The objective was to examine whether words within a given sentence exhibited distinct meanings or senses, and then extract the ones with the closest path similarity. Given that the majority of pun words are nouns, verbs, or adjectives, we deemed it prudent to eliminate stop words and symbols, retaining only the essential information.

However, we soon realized that relying solely on path similarity would yield unsatisfactory results, as it failed to consider the crucial element of context. With the nuance of humor, such context is integral to understanding and interpreting puns. Additionally, stopwords could not be indiscriminately removed, as even seemingly simple prepositions can play a pivotal role in the humor conveyed within sentences.

2.2. NousResearch/Nous-Hermes-Llama2-13b Prompt and Fine-tuning Dataset

To leverage the dataset and collected word senses, we took on the challenge of fine-tuning an existing open-source LLM. Namely, we built upon Nous-Hermes-Llama2-13b, which is itself fine-tuned on over 300,000 instructions on top of the state-of-the-art Llama 2 base model from Meta AI. This model was initially fine-tuned by Nous Research and is available for free download on Hugging Face. This particular model was chosen as it has already been fine-tuned for instruction, thus further instruction fine-tuning on our pun dataset was possible.

The template for the fine-tuning dataset and associated prompting method can be seen in the following example:

```
### Instruction: Use the context to identify which word in the following
sentence is the 'pun word' with the double meaning, it is possible that there
is no 'pun word' if there is no word with two meanings in this context. The
given sentence is 'A scientist doing a large experiment with liquid chemicals
was trying to solve a problem when he fell in and became part of the solution.'

### Response: The 'pun word' is solution. In the context of the given pun,
solution has the two meanings (senses) of:
1. 'a homogeneous mixture of two or more substances; frequently (but not
necessarily) a liquid solution'
2. 'a method for solving a problem'
```

Following this format with appropriate senses from WordNet and our dataset of puns, we performed Parameter-Efficient Fine-Tuning (PEFT) using the AutoTrain Advanced Python library and Google Colab Pro for computation. With this, the same format must be used in order to achieve the best possible performance.

Given that the input was pun, the resulting fine-tuned model correctly identified the ‘pun word,’ the word with the double meaning, in 90.68% of the examples tested from the remainder of our pun dataset. Given that the input was not a pun, the model could generally distinguish a typical sentence such as “How are you?” as not being a pun. However, it performed very poorly if the given sentence was a phrase or proverb such as “Time is money,” and would classify such inputs as puns. The high performance on our testing data, given the input was a pun, was surprising. Fine-tuning seemed to greatly improve responses, even with a relatively small dataset. Still, the limitations of this dataset for such fine-tuning was clear, as in our experimentation, the model had seemingly become biased to very commonly classifying sentences as puns.

2.3. OpenAssistant/llama2-70b-oasst-sft-v10 - Simple yes/no for pun vs no pun

For this usage, we turned to specifically test an LLM for identifying if a given input does or does not contain a pun. The model used is a fine-tuning of Meta AI's Llama2 70B LLM. It is available for free on Hugging Face and was fine-tuned in two stages, first on a mix of synthetic instructions and coding tasks and then in a polishing stage on human demonstrations. This model includes fine-tuning for the usage of a system prompt. The system prompt was utilized for steering the model toward the limited yes/no output.

The prompt template can be seen in the following example:

```
<|im_start|>system
{system_message}<|im_end|>
<|im_start|>user
{prompt}<|im_end|>
<|im_start|>assistant

system_message = "You are a helpful assistant who will be asked to determine if
a given context contains a pun. You will only respond with yes or no"

prompt = f"A pun is a form of word play that uses multiple interpretations of a
word in a given context. The pun word is the word with multiple possible
interpretations in the given context. In the given context, the pun word will
have a more common interpretation and also an alternative interpretation that
relies on a supporting term from a different part of the context. The pun word
is more likely to appear in the latter part of the given context, as there is
often a setup with the supporting term before the pun word. \n For example: Is
this a pun? 'I used to be a banker, but I lost interest.' Simply answer yes or
no: yes. \n For example: Is this a pun? 'Time is money.' Simply answer yes or
no: no. \n Is this a pun? '{sentence}' Simply answer yes or no: "
```

Upon testing this model with our testing dataset mentioned above, it was found to correctly classify pun versus no pun for 73.33% of the examples given. This result was somewhat underwhelming, leading us to explore further methods mentioned below.

2.4. OpenAssistant/llama2-70b-oasst-sft-v10 - Prompt with examples, no format

For this usage, we turned to specifically test an LLM for explaining puns simply based off of a definition and a few examples. We counted using the LLM originally introduced in Section 2.3. We intentionally did not provide a format for output in this case, as we wanted to explore what types of output the model would generate.

Our prompt template can be seen in the following example:

```
A pun is a form of word play that uses multiple interpretations of a word in a given context.
The pun word is the word with multiple possible interpretations in the given context.
In the given context, the pun word will have a more common interpretation and also an alternative interpretation that relies on a supporting term from a different part of the context. The pun word is more likely to appear in the latter part of the given context, as there is often a setup with the supporting term before the pun word.
The following are three examples and explanations of puns in different contexts:
Example 1. : 'I used to be a banker, but I lost interest.'
Explanation of 1. : The pun word is 'interest', which has two interpretations in this context: 'a sense of concern or curiosity' and 'a fixed charge for borrowing money'. In this context, 'banker' is the support term for the pun word 'interest'. The first interpretation of 'interest', 'a sense of concern or curiosity', can be understood even if the word banker is replaced with another job title like engineer. In contrast, the second meaning of 'interest', 'a fixed charge for borrowing money', is supported by the term 'banker'.
Example 2. : 'It's a clumsy reflection of yourself when you break a mirror.'
Explanation of 2. : The pun word is 'reflection', which has two interpretations in this context 'indication of personality' and 'reflection of light in a mirror'. In this context, 'mirror' is the support term for the pun word
```

```
'reflection'. The first interpretation of 'reflection', 'indication of
personality', can be understood even if the word mirror is replaced with
another object like window. In contrast, the second meaning of 'reflection',
'reflection of light in a mirror', is supported by the term 'mirror'.
Example 3. : 'When the nomadic tree senses danger it packs up its trunk and
leaves.'
Explanation of 3. : The pun word is 'leaves', which has two interpretations in
this context: 'exiting a place' and 'leaves of a plant'. The word leaves can be
a verb (the third person singular of leave) or a noun (the plural of leaf ). In
this context, 'tree' is the support term for the pun word 'leaves'. The first
interpretation of 'leaves', 'exiting a place', can be understood even if the
word tree is replaced with another noun like man. In contrast, the second
meaning of 'leaves', 'leaves of a plant', is supported by the term 'tree'.
Using similar thinking, identify and explain the pun word in the following
context by identifying which word can have multiple interpretations given
different parts of the context. Explain the common interpretation in the given
context and then use the supporting term to explain the alternative
interpretation.
New pun: 'Always trust a glue salesman, because they stick to their word.'
Explanation of new pun: The pun word is
```

As seen in the example above, this prompt template was very verbose and potentially difficult for the LLM to digest. Still, given an input that was pun, the model correctly identified the pun word in 85.37% of the examples tested from our testing dataset mentioned above. However, similar to our fine-tuned model, when given an input that was not a pun, it had trouble differentiating puns from phrases and proverbs. It was also not feasible to quantitatively evaluate the explanation, due to the lack of formatting.

2.5. OpenAssistant/llama2-70b-oasst-sft-v10 - Prompted with neatly formatted step-by-step explanation for how to break down puns

For this usage, we turned to specifically test an LLM for explaining puns based off of a definition, the system prompt, and neatly formatted reasoning with Chain-of-Thought Prompting (Wei et al., 2022). Our method for breaking down puns was adapted from Huang et al. (2017) In this section, we continued using the LLM originally described in Subsection 2.3.

The prompt template can be seen in the following example:

```
<|im_start|>system
{system_message}<|im_end|>
<|im_start|>user
{prompt}<|im_end|>
<|im_start|>assistant
```

```
system_message = “You are a helpful assistant who will be asked to determine if
a given context contains a pun. You will think step by step by writing your
steps in determining if a given context contains a pun. After your step by step
response, simply say 'my final answer is no' or 'my final answer is yes'."
prompt = “A pun is a form of word play that uses multiple interpretations of a
word in a given context. The pun word is the word with multiple possible
interpretations in the given context. In the given context, the pun word will
have a more common interpretation and also an alternative interpretation that
relies on a supporting term from a different part of the context. The pun word
is more likely to appear in the latter part of the given context, as there is
often a setup with the supporting term before the pun word.
Now here are some examples of thinking through the questions step by step:
For example: Is this a pun? 'I used to be a banker, but I lost interest.'
Explanation: The context can be broken into two parts of 1. I used to be a
banker and 2. but I lost interest. The word that has two interpretations in
this context is interest. The interpretation of interest related to part 2. is
'a sense of concern or curiosity'. The interpretation of interest related to
part 1. is in relation to the term banking, in this part interest refers to 'a
fixed charge for borrowing money'. So, because there is a word with multiple
interpretations related to different parts of the given context, this is a pun.
My final answer is yes.
For example: Is this a pun? 'Time is money.' There is no word in this context
that can have multiple interpretations, so this is not a pun. My final answer
is no.
Now it is time for the assistant to help:
Is this a pun? '{sentence}' Let's work this out in a step by step way to be
sure we have the right answer."
```

This step-by-step breakdown proved to be very effective in the limited number of experiments we conducted. Due to the random nature of the responses, it was difficult to quantitatively identify how well the model performed in its explanations. Still, in our testing it seemed to outperform our previous models in the interpretation task. Examples of the pun breakdown and interpretation can be seen in section 3.3. The model correctly classified pun vs no pun for 74.26% of the examples given, though this final classification was not always correctly supported by the explanation given. There were also examples in which the model identified and explained a pun sufficiently, but still outputted that the given input was not a pun. This may be due to some ambiguity in the model's understanding of what to consider a pun. Ideas for improving this model are discussed later in Section 5 Fine Tuning.

2.6. Long Short-Term Memory (LSTM) Classification

As previously mentioned, our model exhibits strong performance in identifying pun words and assigning appropriate senses to them. However, it faces challenges in determining whether a sentence qualifies as a pun or not. To address this issue, we employed two machine learning techniques explicitly designed to assess the pun status of given sentences. Thus, this approach can be integrated into the initial stages of our pipeline, prior to the interpretation by our LLM.

Our first pun-detection model is based on the Long Short Term Memory (LSTM) (Hochreiter & Schmidhuber, 1997) algorithm, which was widely used in Natural Language Processing (NLP) tasks before the emergence of transformer models and LLMs. In this case, we constructed a straightforward LSTM supervised classifier that outputs a binary result, indicating whether a sentence is a pun (1) or not (0). To train this model, we utilized the SemEval 2017 dataset. However, it's important to note that this dataset is heavily skewed towards positive labels, meaning it contains significantly more pun sentences than non-pun sentences. If we were to use this data as-is to train our model, it would be inclined to classify a sentence as a pun, which is unrealistic when considering the overall distribution of sentence types.

Although we initially tested the model using the SemEval dataset as it is, we also evaluated its performance with a balanced dataset. This balanced version was created by employing undersampling techniques to ensure an equal number of positive and negative labels.

Table 1 - LSTM models

LSTM model	Positive labels	Negative labels	Accuracy
Biased dataset (original)	2878	1152	~ 84%
Unbiased dataset	1152	1152	~ 82%

The architecture of our model comprises three layers: an embedding layer, a middle layer with 128 LSTM units, and a final softmax layer. Our embeddings were 128-dimensional, and we applied dropout of 0.2 along with recurrent dropout of 0.2. During training, we used the Adam optimizer, a batch size of 32, and conducted training for 5 epochs.

With the biased dataset, our model achieved an average accuracy of 84%, while with the balanced dataset, it achieved 82%. Interestingly, despite a 2-point decrease in overall performance, some specific tests yielded superior results with the balanced dataset. For instance, when evaluating the non-pun sentence "The truck picked up the heavy load," the biased model erroneously classified it as a pun, whereas the balanced model correctly identified it as a non-pun sentence.

2.7. Support Vector Machine (SVM) Classification

Another approach we explored to classify puns was training a Support Vector Machine (SVM) (Cortes & Vapnik, 1995) model for text classification. This was inspired by Rada Mihalcea and Carlo Strapparava in their 2005 paper entitled “Making Computers Laugh: Investigations in Automatic Humor Recognition”, which discussed the results of feature-based classification, a Naive-Bayes approach, and a SVM model with varying accuracy levels based on the type of input text. We decided to test the SVM-based approach, as this method is known for performing well on text-classification tasks, and had reportedly performed best for detecting proverbs, which seemed to be an issue with our original classifier. However, the other forms of classification are not completely off the table, and will likely be tested — alone or in tandem with other models — against our existing classifiers in the future, keeping in mind that the features used in the paper are typical of one-liners, not just puns.

The data used to train our model was a combination of pun and non-pun statements from Reddit's "Humor Detection" Puns by Orion Weller and a list of proverbs from "Proverb Database: Corpus of American English Proverbs (CAEP) and Experimental Study" by Luckert, Claudia. This data is biased, with more non-pun and proverb entries than puns, which made the model more precise towards identifying non-puns than puns (87% compared to 80%). The balance and source of the data is an avenue for further exploration in evaluating the performance of this model.

The classifier has a vectorized, bag-of-words representation of pun and non-pun statements as its X features, and the distinguishing label, 0 for pun and 1 for non-pun, as its y target variables. After tuning hyperparameters and focusing on maximizing precision (as to reduce the number of wrongly classifying non-puns as puns), we settled on the following parameters: C = 10 (less misclassification, harder margin), gamma = 0.1 (intermediate influence of single training values, intermediate curvature of decision boundary when viewed in 2D), and kernel = 'rbf' (radial basis function, projects linearly inseparable data to higher dimensions, classifies based on decision plane). Our current accuracy given these features and parameters is 85.54%, with further testing needed for comparison with our other methods.

3. Results

Our classifiers were tested on three inputs, which were provided by the course instructor. These sentences and our results can be found summarized on Table 2. Additional discussion of classification results follows.

Table 2 - Pun detection using LSTM, SVM, and LLM classifiers.

Input Sentence(s)	**LSTM (unbiased dataset)**	**SVM**	**LLM (step-by-step)**
I am friends with all electricians because we have good current connections.	Yes	No	Yes
I used to play piano by ear, but now I use my hands. It's a sound decision.	No	Yes	Yes
I'm reading a book on anti-gravity, it's impossible to put down.	Yes	No	Yes

3.1. LSTM Classifier Output

With the sole exception of the second sentence, our LSTM classifier accurately classified the remaining two sentences. Perhaps ironically, when using the biased dataset, the model correctly identified the second pun. The most plausible explanation is that when the model encounters a tricky pun, one that deviates from the typical pattern, the biased model tends to lean toward classifying it as a pun, given the dataset's skewed distribution. This is supported by the results from the unbiased model, in which the incorrect prediction fell within a 75% certainty range, indicating a relatively low confidence in the classification.

3.2. SVM Classifier Output

Given the inputs above, the SVM classifier correctly identified the second pun, but incorrectly classified the other two statements as non-puns. We have several hypotheses as to why our model misclassified these statements. The model seems to perform better on statements that only contain one sentence. It was trained to classify statements such as “An apple a day keeps the doctor away” as a non-pun, and “I used to be a banker but I lost interest” as a pun, which could explain its difficulty in classifying inputs with multiple sentences. Our model is also known to be biased towards non-pun statements, having a higher precision in classifying non-puns. Both of these issues reflect the quality and balance of our training data, and we plan to improve our data quality by populating our database with more unique pun statements, as well as search for better and more robust sources for our input statements.

3.3. Step-by-step LLM Output

3.3.1. Test 1

Input:

```
I am friends with all electricians because we have good current connections.
```

Output:

```
Sure, let's break it down step by step:
```

```
1. Identify the key terms in the sentence: "friends", "electricians", "good
current connections"
2. Look for words with multiple interpretations: In this case, the word
"current" could have two meanings. One meaning is related to the friendship
(i.e., "up-to-date" or "recent"), while the other meaning relates to
electricity (i.e., "the flow of electric charge").
3. Determine if the multiple interpretations rely on supporting terms from
different parts of the context: The first interpretation of "current" relies on
the term "friends," while the second interpretation relies on the term
"electricians."
4. Check if the pun word appears in the latter part of the context: In this
case, "current" appears in the latter part of the sentence ("we have good
current connections").
Based on these observations, we can conclude that this sentence contains a pun.
My final answer is yes.
```

Discussion:

In this example, the model is able to break down the pun into sufficient parts, identify the correct word with multiple interpretations, and associate different interpretations of the word current with different parts of the context. The model also correctly identifies that this is a pun. The performance of the model in this example is considered a success.

3.3.2. Test 2

Input:

```
I used to play piano by ear, but now I use my hands. It's a sound decision.
```

Output:

```
Step 1: Identify the different parts of the context.
Part 1: "I used to play piano by ear"
Part 2: "but now I use my hands"
Part 3: "It's a sound decision."
Step 2: Look for any words that have multiple interpretations within the
different parts of the context.
In Part 1, "ear" could have two interpretations - the organ of hearing or the
ability to play music by listening without sheet music.
```

```
Step 3: Determine if these words with multiple interpretations rely on a
supporting term from another part of the context.
The word "ear" in Part 1 relies on the supporting term "by" which links it to
the idea of playing music without sheet music.
Step 4: Decide if the pun word appears in the latter part of the given context.
The pun word does not appear in the latter part of the context.
Conclusion: Based on the analysis above, it can be determined that "ear" has
multiple interpretations within the context and relies on supporting terms from
other parts of the context. Therefore, the answer is:
My final answer is yes. This is a pun.
```

Discussion:

Here, we see that the instructions for breaking down the multiple interpretations are effective, but do not prepare the model for several puns in a single input. In line with our scope, the structure of our step-by-step break down is designed for puns that exploit a single word with two interpretations given different parts of the context. This example is challenging as it has multiple puns involving the words ‘ear’ and ‘sound.’ When two puns exploit multiple meanings associated with multiple words, our method is less effective, as it was instructed to identify only one pun with one pun word and one supporting term. Still, the model both correctly identifies that the input constitutes a pun, and correctly identifies one of the pun words, ‘ear.’

3.3.3. Test 3

Input:

```
I am reading a book on anti-gravity, it's impossible to put down.
```

Output:

```
Step 1: Identify the key words in the context: "reading", "book",
"anti-gravity", "impossible", and "put down".
Step 2: Check if any of these key words have multiple interpretations within
the context:
"reading" -> only one interpretation (the act of looking at written text and
understanding its meaning)
"book" -> only one interpretation (a written or printed work consisting of
pages glued or sewn together along one side and bound in covers)
```

```
"anti-gravity" -> only one interpretation (a hypothetical force that would
oppose gravity)
"impossible" -> only one interpretation (not able to be done or achieved)
"put down" -> two interpretations:
1. to criticize someone or something harshly
2. to place something down, such as putting a book down on a surface
Step 3: Determine if the word with multiple interpretations has a supporting
term that leads to the alternative interpretation:
In this case, "put down" has the supporting term "impossible," which leads to
the alternative interpretation of being unable to stop reading the book due to
its captivating subject matter.
Step 4: Conclude whether the context contains a pun based on the presence of a
word with multiple interpretations supported by the surrounding context:
Since "put down" has two interpretations supported by the surrounding context,
this is a pun.
My final answer is yes.
```

Discussion:

Here, we see that the instructions for breaking down the multiple interpretations are too limiting. In line with our scope, the structure of our step-by-step breakdown is designed for puns that exploit a single word. Yet in this example, the pun is only a single word, ‘down’ when related to the context word ‘anti-gravity.’ It becomes a two-word pun phrase, ‘put down’ when related to the context word ‘book.’

Our model correctly identified the pun phrase ‘put down,’ and the correct relevant sense. However, the model was unable to further break down the phrase to just the word ‘down,’ and instead defaulted to the only other sense of ‘put down’ which it knew of, which was ‘an insult, or something demoralizing.’ It related at least one sense of ‘put down’ to the word ‘impossible.’ Perhaps at some step it inferred that the word ‘impossible’ was demoralizing. However, the output it gave us instead suggested the opposite, and that it believed the obvious and original context was that we were insulting the book, and that the alternative interpretation was that we were captivated by reading it. This could also be a result of instructing the model to look for only a single supporting context term, and to favor less common interpretations.

Our results ultimately show that the model is able to interpret some of the nuance in common phrases, but struggles with the additional steps necessary to associate ‘put down’ with ‘anti-gravity.’ The model correctly identifies the input as a pun.

3.3.4. Test 3b

Input:

```
I am reading a book on gravity, it's impossible to put down.
```

Output:

```
Step 1: Split the context into two parts I am reading a book on gravity and
it's impossible to put down.
Step 2: Identify the pun word(s) in the second part of the context. The pun
word is "put down."
Step 3: Analyze the different interpretations of the pun word. The word "put
down" can mean to place something (in this case, the book) physically lower or
lay it flat.
Step 4: Look for supporting terms in the first part of the context. The
supporting term is "gravity."
Step 5: Determine if the pun word's alternative interpretation relies on the
supporting term. The alternative interpretation of "put down" relies on the
concept of gravity, as something heavy (like a book) might be difficult to pick
up due to gravity.
Conclusion: Since there is a word ("put down") with multiple interpretations in
the given context, one of which relies on a supporting term ("gravity") from
another part of the context, this is a pun.
My final answer is yes.
```

Discussion:

We decided to run one final text using the third input sentence, in which we removed one aspect of negation from the sentence. We changed the word ‘anti-gravity’ to ‘gravity.’ After editing, the model does identify ‘gravity’ as the supporting term for one interpretation of ‘put down.’ However, the explanation it gives for the relevance of ‘gravity’ is not exactly what we would expect. This could be a result of the pun no longer making sense, as gravity would make a book harder to pick up, not to put down. We find it significant that the model succeeded in identifying ‘gravity’ when it could not identify ‘anti-gravity.’ This could possibly be because ‘anti-gravity’ was a less common term than ‘gravity’ in its original training set, and that the model struggled to closely associate ‘anti-gravity’ with the opposite of whatever terms ‘gravity’ would otherwise be associated with. The model continues to identify the input as a pun.

4. Conclusions

The three input sentences provided by the course instructor posed an interesting challenge for our program. One of the three sentences arguably contained more than one pun: “I used to play piano by **ear**, but now I use my hands. It’s a **sound** decision.” Here, ‘ear’ invokes two word senses, which are ‘to play without instruction or written music’ and ‘an organ one hears with.’ However ‘sound’ also invoked two word senses, which are ‘audible vibrations’ and, ‘good, solid, dependable.’ Our program was only capable of detecting a single pun word at a time, and we would expect it to be likely to select either ‘ear’ or ‘sound.’

Another input sentence featured only a single pun, but the pun encompassed two words: “I am friends with all electricians because we have good **current connections**.” In this example, ‘current’ meant both ‘recent’ and ‘flowing electricity,’ but ‘connections’ meant both ‘interpersonal connections’ and ‘physical connections between plugs, sockets, and wires.’ Our program was only prompted to detect a single pun word, and thus we see it select ‘current’ in this case.

The final input sentence posed a challenge that appears to have been problematic for natural language processing applications: negations. This pun was, “I’m reading a book on anti-gravity, it’s impossible to **put down**.” In this sentence, both ‘gravity’ and the possibility of ‘putting something down’ have been negated. Furthermore, the pun word is actually a phrase. This phrase is ‘put down,’ which has an additional sense of ‘discontinuing the reading of a book’. As our models were usually looking for a single pun word, and are designed with specific limitations regarding structure and use case, we expected our program to have mixed results with such inputs.

Given the challenges posed by all three inputs, we were delighted when our different models were able to recognize the puns, and not surprised by the shortcomings. To correctly identify sentences with multiple puns, we would have to structure and train these models differently. To successfully handle negation and phrases, we would need to employ solutions specifically tailored to that end, which in itself serves as a substantial undertaking. Still, we were pleased and impressed with the performance of our models, both inside and outside of the use cases we designed them for.

5. Future Work

There are still aspects of our program which we would have liked to improve. To start, we acknowledge that our database of non-puns for training and testing could use improvement. The proverbs database had endeavored to model a plurality of puns versus non-puns that often emerged from a similar stem or patterns. For the purposes of training, it is arguable that these could be considered duplicates, and that the dataset must be cleaned of them. In addition, some proverbs are themselves puns, or at least engage in some form of wordplay. We did not manually prune the database of such cases. Improvement of reasoning and interpretation capability might also be attained through the use of similar datasets to SemEval-2017's, such as those used in the Evol-Instruct method (Xu et al., 2023).

Furthermore, in order to train a model on our SemEval-2017 dataset, we were required to reference WordNet word senses to make interpretations. These word senses are extensive but not exhaustive and cannot be used to explain every pun. This reliance on WordNet may have limited our fine-tuned Nous-Hermes-Llama-2 (13B) model. Despite that, there are no easy alternative ways to evaluate whether an LLM has correctly interpreted a pun.

We could try to improve pun interpretation by fine-tuning our OpenAssistant LLaMA 2 SFT v10 (70B) model, but this LLM was prompted to generate step-by-step breakdowns to explain puns. To train and test it, we would need a dataset that contained step-by-step breakdowns, not WordNet word senses. After finding or building such a dataset, we would then have to determine the best way to evaluate when the model had correctly identified the right steps, and how to calculate loss, which in itself may present a natural language processing hurdle. Additionally, we presently lack the physical computational resources to fine-tune this model, which is an order of magnitude more complex than the Nous-Hermes-Llama-2 (13B) model.

References

Cortes, C., & Vapnik, V. (1995). Support-vector networks. *Machine learning,*, *20*(273-297).

Hochreiter, S., & Schmidhuber, J. (1997). Long Short-Term Memory. *Neural computation*, *9(8)*, 1735-1780. IEEE.

Huang, Y.-H., Huang, H.-H., & Chen, H.-H. (2017, April). Identification of Homographic Pun Location for Pun Understanding. *WWW '17 Companion: Proceedings of the 26th International Conference on World Wide Web Companion*, 797–798. 10.1145/3041021.3054257

Lückert, C. (2019). Proverb Database: Corpus of American English Proverbs (CAEP) and Experimental Study. doi 10.17879/14159451387

Miller, T., Hempelmann, C., & Gurevych, I. (2017). SemEval-2017 Task 7: Detection and Interpretation of English Puns. In *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)* (pp. 58-68). Association for Computational Linguistics. 10.18653/v1/S17-2005

Wei, J., Schuurmans, D., Bosma, M., Ichter, B., Xia, F., Chi, E. H., Le, Q. V., & Zhou, D. (2022). Chain-of-Thought Prompting Elicits Reasoning in Large Language Models. *Advances in Neural Information Processing Systems (NeurIPS 2022)*, *25*, 24824-24837. 10.48550/arXiv.2201.11903

Weller, O., & Sepppi, K. (2019, November). Humor Detection: A Transformer Gets the Last Laugh. *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, 3621–3625. 10.48550/arXiv.1909.00252

Xu, C., Sun, Q., Zheng, K., Geng, X., Zhao, P., Feng, J., Tao, C., & Jiang, D. (2023, April 24). WizardLM: Empowering Large Language Models to Follow Complex Instructions. *arXiv preprint*. 10.48550/arXiv.2304.12244