A True Daily Double: Historical Findings and Predictive Modeling in Jeopardy!

Dennis Smith, Amar Yadav, Shijia Zhou

Abstract

Using a dataset of over 200,000 Jeopardy Questions and Answers, we investigated historical insights surrounding these questions and made predictive assumptions. K-Means Clustering and T-SNE were used for historical analysis, and CBOW (Complex Bag of Words), Skip-gram, and Llama 2 were used for predictive modeling. Historical analysis showed specific topics are preferred over common questions, and “easy” questions were the most common. Our predictive modeling approaches developed sound methodologies but were ultimately constrained by the processing capacity of our machines.

Background

Before our attempt at Question Answering (QA), we wanted to understand the Jeopardy Dataset. To do this, we chose K-Means sampling to group similar data points by their features, and t-SNE (t-Distributed Stochastic Neighbor Embedding) to improve our understanding of the relationships between questions and topics. For the Jeopardy Dataset, we hoped to use K-Means sampling to cluster questions by topic, difficulty, and category. Combining this with t-SNE would visualize the similarities and differences in the questions asked at each difficulty level, and help us better understand the data when testing our QA implementations.

Early QA approaches used rules-based models, which relied on grammatical semantics and functioned like a decision tree. These required models to be closely tailored to the corpus, as grammatical nuances vary greatly between textual components. Statistical approaches followed, which improved the variability needed in a QA model. However, these models required a hypothesis to set the model's tone, and still underperformed in adaptability and the ability to handle the variation of language (Ishwari). The introduction of Machine Learning (ML) and Artificial Intelligence (AI) led to more dynamic QA models capable of creating and building their knowledge base. These models can train and test with greater independence but still require careful programming and domain expertise(Ishwari).

We considered David Elworthy’s *Question Answering using a large NLP System* for design and architecture considerations. Microsoft created a model using a retrieval engine, their own NLPWin NLP system, and a form-matching module. The architecture used a Question Analysis stage to generate logical forms and extract query terms, Information Retrieval to list relevant documents and segment them into sentences, Sentence Processing to order sentences by query terms, and Logical Form Comparison to create a list of answers with corresponding scores (Elworthy).

Our approaches - Skip-gram, CBOW, and Llama 2 - build on ML by incorporating deep learning and neural networks. These networks learn the underlying features of the data through training and have been shown to outperform all previously mentioned approaches. Skip-gram and CBOW follow the Word2Vec Framework and use shallow neural networks to train word embeddings, while Llama 2 uses a deep neural network architecture based on transformers. These transformers help capture long-range dependencies in text data. Additionally, Llama 2 follows some of the system architecture discussed by Elworthy, where we extract terms using question analysis, process these sentences, and finally create a list of possible answers to the original question.

Method

K Means/ T SNE Method

The data were initially cleaned up by removing missing values and other irrelevant information. To simplify the process of evaluating question difficulty, a "Difficulty" column was added and all questions were categorized into easy, average, or hard based on the round they belonged and their money values. Specifically, questions from the Jeopardy! Round valued at $2000 or less were labeled as 'easy'. Questions valued above $2000 in the Jeopardy! Round or up to $3000 in the Double Jeopardy! Round were classified as 'average'. Questions from the Double Jeopardy! With a value higher than $3000, the Final Jeopardy! Round and any tiebreakers were considered 'hard'. Next, text processing used stopwords, wordNetlemmatization, and setting everything to lowercase.

Questions and categories were vectorized using the TF-IDF (Term Frequency Inverse Document Frequency) method. In other words, I've transformed the raw text from the Jeopardy dataset questions and categories into a matrix whose entries are the numerical features representing each word in the text. This feature extraction focused on the most relevant 1000 features. Next, the naturally ascending order of the difficulty levels was mapped. All features were combined via hstack into a single feature matrix. After standardizing the features, PCA reduces dimensionality without missing 95% of the total data variation. We conducted the elbow method to find out the most effective number of clusters. Then, followed by K-Means clustering implementation. For visualizations, we incorporated t-SNE to visualize the data. Due to the computational limit, I selected a 10000 random sample to speed up the t-SNE process.

Skip-gram CBOW Method

The goal was to classify the jeopardy questions based on the categories. This started with preprocessing the data to remove stopwords, standard tokenization, and removal of any character that is not lowercase or uppercase. The preprocessed questions are then pairs of the questions themselves and the category. For each preprocessed question, embeddings for the individual words are retrieved and averaged to represent the entire question. This is done using the generate_skipgram_embedding function, which iterates through each word in a question, checks if it exists in the trained model's vocabulary (model.wv.key_to_index), and then aggregates these vectors to compute a mean vector. If no words from the question are found in the vocabulary, a zero vector of the same dimensionality is returned. Both Skip-gram and CBOW models are used in the same way to generate embeddings.

After generating these embeddings, they are used as features for training logistic regression models. Separate models are trained using Skip-gram and CBOW embeddings, respectively. This process involves splitting the data into training and testing subsets, fitting the logistic regression model on the training data, and then evaluating its performance on the test set.

The utilization of both Skip-gram and CBOW models allows for a comparison between the two methods regarding their effectiveness in capturing semantic meanings of words within the context of Jeopardy questions. This comparative analysis can help determine which embedding technique is more suitable for natural language processing tasks in different scenarios.

QA Algorithm Method

Originally, we planned to implement a BERT (Bi-directional Encoder Representations from Transformers) model to implement a modern QA algorithm. BERT considers context from both directions in a sentence to better understand word nuances and dependencies. It can also be fine-tuned for question answering and be pre-trained. However, BERT's primary purpose is understanding and processing text, rather than generating a coherent response. This did not meet our expectations for a QA system that could attempt to answer Jeopardy questions.

We then discovered Llama 2, which runs on GPT-3. Llama 2 is generative and is valuable for human-like responses and context generation. It also has a large content window for longer questions. For a quiz-style question, Llama 2 is particularly useful as it can make guesses with a few examples (few-shot learning) or zero examples (zero-shot learning). It has a large pre-training dataset that allows us to attempt question answering without pre-training our algorithm on the extensive dataset required to answer Jeopardy's varied questions.

I used Hugging Face for the transformers, PyTorch for deep learning operations, and Accelerate to optimize PyTorch operations. For training and tuning, I loaded Bits and Bytes and PEFT (Parameter efficient fine-tuning) to optimize the speed of the LLM, and QLoRA (Quantized Low Rank Adapters) for further fine-tuning. QLoRA uses a small number of quantized, updateable parameters to limit training complexity (Culliton).

The next step involved pulling our data, the 200,000+ Jeopardy Questions and answers. I created a data frame, which I later used to choose a prompt value. This prompt would ask Llama 2 the particular Jeopardy question I wanted to be answered (Culliton).

Configuration and Tokenization began by implementing BitsandBytesConfig(), which loads the weights in a 4-bit format with double quantization to improve QLoRa's performance. I also added the Auto Tokenizer at this stage (Culliton).

prepare_model_for_kbit_training() is a PEFT wrapper that will help us train and tune the model. It sets the output embedding layer so that we may later provide gradient updates, and perform typecasting on components required for updating the model. Helper functions below for layering will later help us identify the updated layers (Culliton).

Configuring LORA began using LoraConfig(). *r* is the width of the small update layer. The goal is to set this layer wide enough to capture our problem's complexity. The simpler the problem the smaller *r* can be set (Culliton).

The next step is the generation process. Temperature uses the softmax function to determine output values. Top P serves as a method for choosing from a selection of outputs. Finally, we'll limit return sequences to 1 as we want one answer for a given question in Jeopardy. Finally, we can run a function to run the algorithm (Culliton).

This untrained version does not provide a reasonably informed answer. Therefore, our model requires training. I'll use the HuggingFace transformers library to create a training loop. This process will make a single pass over all of our data. The process to train over the full dataset was projected to take 16+ hours, so we unfortunately had to run this data on a subset that was 1% of our total dataset (Culliton).

Results

K Means/ T SNE Results

The t-SNE visualization is more than just a good illustration of the mix of our Jeopardy question data given that it highlights a range of random topics. The specific color code for each cluster has its meaning and gives a vibrant image of interlinked clusters instead of isolated beings. This exquisite pattern helps us to elaborate the various layers of the relationship of different themes harmoniously so well in the show and how knowledge domains are closely interconnected. Importantly, clusters 0 and 1 come out as the major entitles situated at the dense area of such graph, which is intensively connected, suggesting a small number of questions about common subjects. On the contrary, there are groups of clusters (eg. 7 and 3) that show higher dispersion indices, meaning they most likely are concentrating on more specific and discerning topics.

The analysis of the clusters' sizes and the distribution of question difficulty within each group was followed up. The assessment would imply that the biggest cluster includes 138,465 questions, with 52% falling in the "easy level". Cluster 3 shows the dominating of those questions that are assessed as average in complexity. Although it is smaller, cluster 8 has a large number of "average" and "hard" questions, with the highest ratio of "hard" questions observed across all clusters indicating a possibility of subject focus on more specialized topics or complex topics for the data set group. None of the clusters discussed oftentimes have a characteristic to tackle issues considered to be rather hard.

Skip-gram CBOW Results

Due to the limitations of the local machine, a slice of the training dataset was used to generate embeddings and train the models. It took 4 hours for both the Skip-gram and CBOW model but the accuracy scores were not satisfactory. There are several reasons as to why that could happen as discussed in the section below.

QA Algorithm Results

The original attempt for Llama2 involved pipelines and a response to a direct question. For example “For the last 8 years of his life, Galileo was under house arrest for espousing this man’s theory” should be answered with “Copernicus”. However, the model would respond with open-ended questions, asking “What were Galileo’s major achievements?” or “How was Galileo’s life changed by Galileo’s arrest?”

The next iteration of the algorithm is the process I described above, which involves training and tuning the model in the hopes of improved results. Unfortunately, the model would not respond to these prompts, likely because the computational limitations of my machine prevented the model from gaining more expansive training.

Model perplexity was infinite due to the training limitations. While the implementation focused on efficiency, there is only so much to be done to mitigate the processing of 200,000+ entries. I’m confident that we would see improved perplexity and some accurate QA without the reality of our processing limitations.

Discussion

K Means/ T SNE Results

The data clustering and visualization achieved through t-SNE have enabled us to comprehend these complex relationships among the questions in Jeopardy. Many groups of clusters overlap, due to many questions having components exploring different subjects or fields of knowledge. The t-SNE visualization technique provides a vivid overview of the Jeopardy dataset, but does not provide an ideal categorization, nor does it clearly distinguish between different clusters. For future work maybe consider implementing Word2Vec instead of TF-IDF and using the latent Dirichlet allocation (LDA) model to perform a similar analysis for comparison.

Skip-gram CBOW Results

Jeopardy questions cover a wide range of topics, from history and literature, to science and popular culture. The complexity and diversity of the content can make it difficult for a model to accurately classify questions, especially if the categories are broad or overlapping. There were 28,000 unique categories for 200,000 questions which gives a 1:7 ratio of questions per category. Further, the Word2Vec embeddings used in the model may not capture enough semantic meaning to differentiate between categories effectively. Word2Vec provides a dense representation of words based on their context, while Skip-gram and CBOW models are good at capturing syntactic similarities. However, these models are less effective at capturing deeper semantic meanings needed for accurate classification. Averaging word vectors to represent entire questions can also dilute important signals. Keywords that might be critical for classification can lose their impact when averaged with less informative words.

QA Algorithm Results

Llama 2 showed a slight response before training, showing that we can get a response from the system. Unfortunately, the processing constraints of our systems prevent us from training on the full dataset, which would allow us to more effectively answer Jeopardy questions. The process and algorithm appear effective but are unfortunately held back by processing capacity.

Conclusion

Our historical analysis findings centered around topic difficulty and specificity. Jeopardy topics tend to be more specific rather than a common question, which is fitting from a category-based quiz show. Most of the questions tended to be graded “easy” as the game ramps up from a variety of easier questions before increasing in difficulty and decreasing in question quantity.

Our predictive modeling was marred by processing constraints, which prevented the models from being trained on the entire dataset. In the future, we look to deep learning models like LSTM or GRU networks rather than Skip-gram and CBOW. These models can capture the order of words in a text, a feature that could be particularly beneficial for question classification. Another improvement could be combining the predictions from multiple models to improve accuracy and robustness. For example, ensemble methods like Random Forest or Gradient Boosting could be tested in addition to logistic regression.

Finally, a transformer model like Llama 2 would be invaluable as a generative QA algorithm but requires greater processing capability. This would lend to better training and processing of large datasets that make these generative models effective.

Works Cited

Culliton, Phill. “Fine-Tuning with Llama 2 + QLoRA.” *Kaggle*, Sept. 2023, www.kaggle.com/code/philculliton/fine-tuning-with-llama-2-qlora/notebook.

Elworthy, David. *Question Answering using a large NLP System*.

Ishwari, K.S.D, et al. *Advances in Natural Language Question Answering: A Review*.

Nathan, Aravind Ram. “Jeopardy Dataset (Updated)” *Kaggle*, Apr. 2022, https://www.kaggle.com/datasets/aravindram11/jeopardy-dataset-updated.

Tomas Mikolov, Kai Chen, Greg Corrado, Jeffrey Dean, Efficient Estimation of Word Representations in Vector Space.

Research Contributions

Dennis Smith - Llama 2 Modeling

Amar Yadav - Skip-gram and CBOW Modeling

Shijia Zhou - K-Means Sampling and t-SNE Distribution Analysis