Understanding Jeopardy Valuation Through Natural Language Processing

Seth Singson-Robbins
Fordham Univesrity

ssingsonrobbins@fordham.edu

Abstract—This paper investigates the valuation of Jeopardy designing a model that could correctly classify both the round and value that Jeopardy answers should be in. This requires a great deal of Natural Language Processing and utilizes word embeddings, Neural Networks, and other Machine Learning models to determine that best predicts the classes these Jeopardy questions were assigned. The results of the final model utilized had an accuracy of 15.2% accuracy, much higher than random guessing which would have been 10.3%. This investigates additional analysis of how the model performed and questions validity of the valuations of Jeopardy questions which have the final class determined by the head writer of the show.

I. Introduction

Jeopardy is a prime-time gameshow that started in the United States and, according to their website, started in 1964 but has been continuously running since 1984. The gameshow has been awarded 34 Emmy awards [(1)] with versions of the show having versions of the game in over 25 countries according to liveabout.com [(2)].

The gameplay involves 3 contestants who pick dollar values on the board that flip over to show a question in the form of an answer. They have clickers in their hands and the first to press their clicker has the chance to the answer in the form of a question in the form of a wager. If they get it right, they are awarded that money and it goes into their bank that is displayed in front of their podium. If they get it wrong, they lose the equivalent amount of money from their bank.

The game is divided into 3 rounds. The first round is called Jeopardy which consists of 5 categories each with 5 problems and have a range of values ($100 and $500 between 1984-2001, $200 to $1000 from 2002 onward). They select values on the board and receive the problem in the form of an answer and must respond to the answer in the form of a question to get the problem correct. Double Jeopardy has a similar structure but the valuations double in value ($200 to $1,000 between 1984-2001, $400 to $2000 from 2002 onward). In both rounds they have hidden answers called 'daily doubles' where they can wager up to the value of either the highest value of the round or what is available in their bank before seeing the answer. Final Jeopardy is the last round where contestants wager up to the value they have banked before receiving the problem. If they have a bank value at $0 or below, they will not be able to participate in Final Jeopardy. The contestant with the highest amount of money accrued wins the game, rewarded all the money they banked, and is invited to play again the following day.

In terms of the formulations of the answers including their valuation and category placement, the Jeopardy's article about the writers' room [(3)] explains that there is 8 writers and 7 researchers who work on the show coming up with 14,030 answers and questions per year. The writers determine what will be displayed on the board with the head writer determining if they are good enough to be on the show along with the rounds and values ensuring the answers increase in difficulty. One strategy they try to ensure is that at least one person can answer the question but not too many questions can be answered by everyone in the room. Researchers then come in using the most accurate sources utilizing resources like official encyclopedias to ensure that the questions and answers are accurate.

This paper will be trying to determine if there is a way to utilize Machine Learning and Natural Language Processing techniques to help predict what the round and valuation of a Jeopardy answer and question would be and, if so, some analysis of what goes into the difficulty of an answer. For simplicity, the rest of the paper will call the 'answers' the problems and the 'questions' the solutions.

II. Methodology

This section looks at the additional work done to determine and run the model. This includes how the data was collected, some dataset exploration, preprocessing on the data, training versus test data, tokenization, embeddings used, and what models including hypermeters were tested for final evaluation.

A. Data Collection

J-Archive [(4)] is a website that has recorded every single Jeopardy problem with additional information since its 1984 prime time debut. This data was converted into a JSON file and posted by Bojan Tunguz on Kaggle and was used for this analysis. [(5)] It has a total of 216,930 problems and solutions from the game between the years 1984-2012. The helpful features for analysis include the problem which is called text, and solution (not in question form) which is called answer, the round, and the value or wager of the contestant. Additional information includes the problem category, the airdate, and show number.

B. Data and Text Preprocessing

Given the nature of the data, there were several modifications that needed to be done to the data to be ready for analysis using the pandas [6] and re packages. [7] The data was shuffled to minimize time-based bias due to being in chronological order. Data was also changed to match datatypes that match their actual values (dollar signs removed to make it a float, air date changed to datetime type, etc.). The year was also separated into a new column for additional preprocessing and analysis explained later.

For the actual values used for classification, there were 11 values assigned based on the round and value. Round named were changed to be shorter (Jeopardy! was transformed into J!, Double Jeopardy! into DJ!, and Final Jeopardy! into FJ!). Since the values of the game doubled abruptly in 2002, values in 2002 and later were halved to ensure that there is equivalent analysis. We will be looking at if the increase in problem values in 2002 had an effect on the results later in the paper. Final Jeopardy not having a value had values set to 0. Rounds and values were then concatenated to form the final 11 classes used for analysis.

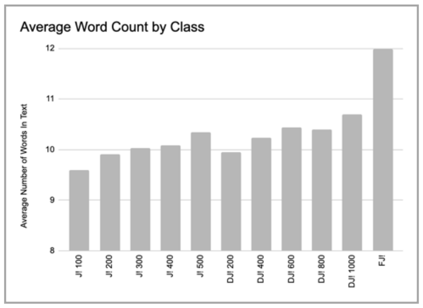
Daily double questions were removed from the analysis due to the problems showing their wager rather than the actual value on the board creating noise that could hinder the results.

The actual text had modifications to keep the text consistent and maximize its effectiveness in the models. All contractions were split up so that the actual words are all shown. An additional column that shows each word in an array was created for every text since several of the models require this for analysis. Some of the text linked to pictures or videos so those were all removed and changed to just 'http' to signify that there was a picture or video in that spot. Only words were kept and anything that is considered a stop word by the NLTK corpus [8] was removed from the word list. Lastly, each problem and solution were concatenated into a final word list and text. The results section will also look at what the performance would be if only the problem was utilized.

C. Data Exploration

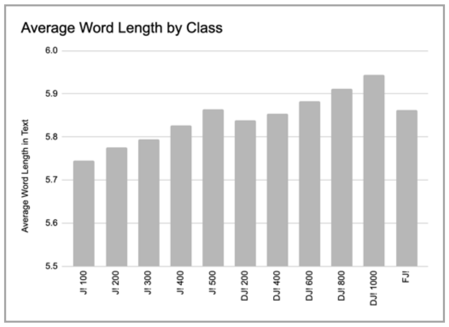
It is worth noting a few insights about the problem text and its relationship with the problem value. The average word count per class in Figure 1 has a relationship where, the larger the value, the larger the average number of words are. This increases further the later the round with Final Jeopardy! being much larger with an average of 12 words versus 9-11 words for the previous rounds.

Figure 1. Average Word Count by Class



The average length of a word show in Figure 2 has a similar relationship except the Final Jeopardy! round not standing out like Figure 1 but being around the same as Jeopardy! $200 and Double Jeopardy $400. Note that these insights are averages and still have large variances within each category so they cannot alone determine the category and value but highlight that there is a relationship in an aggregate sense.

Figure 2. Average Word Length by Class



When looking at the most common words, there are some interesting findings. Figure 3 shows the top 10 words in each of the categories. Links from pictures and videos are commonly used (shown as http) so will not be a determining factor in dividing the categories at least as a stand-alone word (could have an effect in relationship to the sentence). Other words common in every category also include the number one, first, and city. Words that are specific to certain categories include the word state only being in the top 10 for Jeopardy! 100 and Jeopardy! 200. The word clue is only common in Double Jeopardy! 600 and Double Jeopardy! 1000. The US, the word world, and 1 in numerical form being in the text is only most common in Final Jeopardy! These could be contributing factors in determining the valuations of each category.

Figure 3. Top words in each class

Top Words Rank	J! 100	J! 200	J! 300	J! 400	J! 500	DJ! 200	DJ! 400	DJ! 600	DJ! 800	DJ! 1000	FJ!
1	one	one	one	one	name	one	one	http	one	http	2
2	br	br	br	br	br	first	br	one	br	name	first
3	first	http	http	name	http	br	http	name	name	br	name
4	name	first	name	http	one	name	name	br	http	one	one
5	city	name	first	first	first	http	first	first	first	first	U.S.
6	http	city	city	city	city	city	city	city	city	city	man
7	2	2	new	2	2	country	country	clue	man	clue	named
8	state	new	2	named	named	new	2	country	2	seen	1
9	new	country	seen	new	seen	2	man	man	country	named	world
10	country	state	man	seen	country	man	new	2	named	called	city

D. Training and Evaluation

Several models and methods to set up the data were necessary to ensure that the optimal model was utilized for the final results. To help with the evaluation of the numerous models, 20% of the dataset was separated from the training data to help with the final evaluation of all the models. Of the remain training data, An additional 20% was set aside to be used for training validation to ensure that the model and hyperparameters chosen had the highest accuracy before being utilized for the final test data and results.

E. Tokenization

For the models that utilize word lists, tokenization is required for a mapping and organization of the data which will be transformed into different quantitative representations via embeddings allowing the models to read the data. Tensorflow has the Tokenizer() class [9] that turns every word into a number and has a mapping available to reference the word. This transformation and mapping were fit onto the feature training data. Due to text being of varying lengths, padding was also required to ensure that the array lists were all the same size with each text being as many items as the longest text at 88 words and shorter texts having zeroes after the last word.

A table that signifies which class the text is classified in also needs to be in a form that the machine learning models can read. SkLearn has a LabelEncoder class [10] that transforms the data so that each class is a field in an array. This is binary so every item in the array will be 0 except for the item in the array that represents the text is categorized under and has a value of 1. The data is categorized so that it has an array with a shape of the number of examples and 11 for each of the classifiers that it could possibly be labeled under. A class encoder is saved to transform them back into their respective classes for final results and analysis.

H. Embeddings

Text embeddings are required to create a numerical structure that represents either the word or the text that the machine learning models can understand. Four different methods were used with validation data being used to pick the best embedding model.

Term frequency-inverse document frequency is a method that looks at how unique a word in the text is in relation to other texts in a dataset. This helps identify what makes a document unique and what words are important for separating different documents in different classes.

GloVe is an embedding model created at Stanford University [11] and creates a vector representation of a word based on the frequency of words in the dataset. Like TdIdf, the model looks at the frequency of words in a text compared to other texts in the dataset. The GloVe model being used creates a 50-item vector for each word which creates a 2D representation of the text for each example in the dataset.

Word2Vec is a word embedding model created at Google and uses the co-occurrence of words within the text of each example to create an 2D array which will be a of every word having a 100-item representation. Gensim [12] will be utilized for this embedding.

Bidirectional Encoder Representations from Transformers, also known as BERT, is a neural network also created by Google that utilizes transformers and bidirectional encoding (hence the name) to create a numerical representation of the data. The models that utilized BERT were pulled from Tensorflow Hub [13] and are called Small-Bert having 4 layers and result in a 128-item numerical vector for each of the text in the dataset. Given that BERT analyzes the sentence instead of looking at the relationship of each word, the actual text is utilized for embedding instead of the created word lists like the other embedding models.

F. Models and Hyperparameters

For the actual machine learning models used, there were 5 distinct models used to find the best performing model. Each model has its pros and cons, and the performance of the models can help reveal insights into the relationships of the data with the value classification. Models used four embeddings listed in the previous section when applicable.

Three of the models used were neural networks powered by TensorFlow. [9] The first model is a general neural network that has hidden layers and dropout to help minimize overfitting. The second model is Recurrent Neural Networks (RNN) that models the data using Long Short-Term Memory (LSTM) and works for models that have a sequential patten, something common in Natural Language Processing tasks. Given the sequential nature of the neural network, TfIdf was not applied as an embedding to this model. The third Neural Network was Convolutional Neural Networks (CNN) which looks at the neighbors around each word in the text to help group and find patterns in the text. This typically is utilized for image recognition but can also be useful in Natural Language Processing tasks when the nearby neighbors have a high influence on the classification. All three neural networks were tested for performance with the same hyperparameters: number of epochs, number of hidden layers, number of items in each hidden layer including if equal or different by layer, and value of the dropout.

The other two models tested were utilizing SkLearn's machine learning package. [10] The first was Native Bayes which looks at the likelihood of being in class given the value of each item in the dataset. This model if strongest would highlight that it is the words rather than the order or relationship of the words that is important and that all words have equal influence in the results. Multinomial Naïve Bayes was used since this a multiclass model and is frequently used in Natural Language Processing models. The hyperparameters used were if prior probabilities were used in the model and the smoothing parameter strength (alpha). The other SkLearn model utilized was logistic regression which creates a weighted numerical model to determine the likelihood of each class. Like Naïve Bayes, this has independent relationships of each word but, due to the weights of each text, the spot of each word can have a different amount of influence on what value the text is classified under. Multinomial was utilized to minimize the loss across all categories. The hyperparameters used the regulation strength (C), if l2 was used for smoothing, and the solver.

III. Results

The results of the model have been evaluated by looking at each model's training performance on the validation set with the best model utilizing the test set to look at the results. The categories were mostly equal, except for Final Jeopardy!, so accuracy was used for performance purposes. If the model predicts the correct category, that counts as a win with the wrong category being a loss. Precision and recall were utilizing for a deeper dive in the results. The class with the most representation in the test data was Jeopardy! $100 at 10.3% of the dataset so that will be used as a baseline as the assumption is that model should at least perform better than selecting the most common class.

Figure 4 below shows the accuracy of each embedding and model on the validation data. The RNN model utilizing GloVe for its embedding had the strongest performance with the optimal parameters achieving an accuracy of 15.5% on the validation data. GloVe performed the best compared to the other Neural Networks but did not perform as well on the other Machine Learning models. RNN also performed the best for each embedding except for BERT (logical since BERT returns an array based on the entire text and is not as sequential) and TdIdf which was not utilized for RNN modeling.

Figure 4. Accuracy by Embedding and ML Models

Model	GloVe	Word2Vec	BERT	TdIdf
Typical NN	14.2%	13.9%	13.7%	12.6%
RNN	15.5%	14.8%	13.0%	NA
CNN	13.8%	13.1%	13.7%	12.8%
LR	11.4%	12.0%	12.2%	12.4%
NB	13.3%	13.0%	12.3%	13.4%

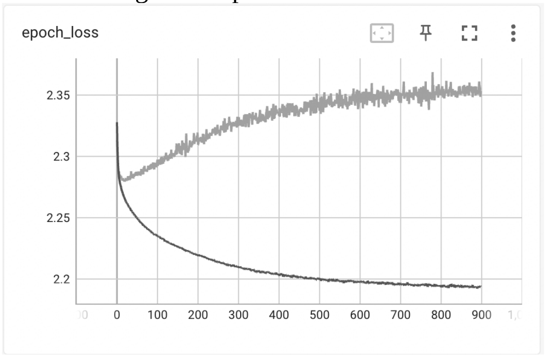
When applying the model on the test set, there was a slight drop with an accuracy performance of 15.2%. While this is not optimal and could not identify most of the dataset, it is still a 47% improvement over random change showing that it could find some patterns in the data. Figure 5 shows precision and recall of each category for a deep dive into the results. Precision was low across the board while recall stood out for Jeopardy! 100 and Double Jeopardy! 400.

Figure 5. Precision and Recall by Class

Category	Precision	Recall
Jeopardy! $100	16%	44%
Jeopardy! $200	12%	8%
Jeopardy! $300	12%	8%
Jeopardy! $400	11%	24%
Jeopardy! $500	14%	11%
Double Jeopardy! $200	19%	37%
Double Jeopardy! $400	14%	25%
Double Jeopardy! $600	12%	9%
Double Jeopardy! $800	11%	4%
Double Jeopardy! $1000	13%	1%
Final Jeopardy!	24%	13%

Additionally, running this model for additional epochs can lead to additional insights of the data. When increasing the epochs indefinitely, if the loss from the training data does not converge to 0 then it is likely due to the data not being linearly separatable and that the model cannot find enough patterns to be able to separate the dataset. Figure 6 shows the model utilizing Tensorboard from Tensorflow [9] with 900 epochs with the lower line being the training data and converging well above 0. The validation data above is also starting to increase showing overfitting happened early very early on (the final model for the test data only utilized 18 epochs due to being the maximum accuracy on the validation data). With three hidden layers, this shows that there are some patterns in the data but not enough with that many hidden layers to accurately separate the dataset.

Figure 6. Epoch loss overtime



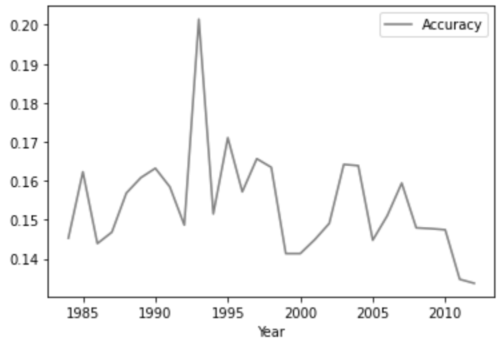
With RNN and GloVe being the most accurate, it could reveal that the data is most accurate when looking at the data in a sequential manner and has a connection with the relationship between each word. It was surprising also that BERT was not the model accurate model but, due to utilizing transformers and other complicated modeling, this could be due to being too complex leading to more overfitting than GloVe.

IV. Additional Analysis

With the test model there are several insights that one could ask of the data. Given the low accuracy, these should be taken as not guaranteed to be the case and stronger models could reveal conflicting results.

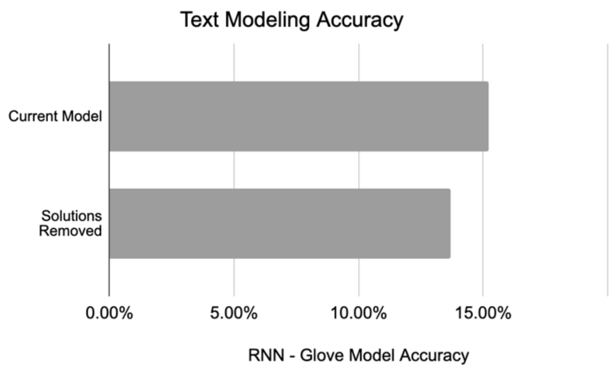
Given that the value of the categories doubled in 2002 and effects like inflation, one could ask if the accuracy of the model had any directional patten with the year the problem occurred in the game. If it is the case, one could also ask if modeling by year could reveal additional insights such as that the game itself is getting easier or harder by seeing if you placed an earlier year problem in a model that was in a later year if the categorization was for a higher or lower valued category. However, this is not the case as shown in Figure 7 which shows that the performance is variable and does not show any obvious case for this relationship.

Figure 7. Accuracy by Year



Additionally, Figure 8 shows how the model would have performed if the solutions were removed from the text instead of being concatenated in the model. The performance was lower across the board but for this model the performance is above a percentage point lower.

Figure 8. Performance vs Without Solutions



V. Related Work

There has been similar analysis to the Jeopardy dataset and predicting values from other people across the internet.

Rajarshi Ghoshal [14] showed his study on his Kaggle page predicting if the question is in Jeopardy, Double Jeopardy, or Final Jeopardy using Random Forests.

Awisk [15] has a medium post where they utilize BERT and other modeling to predict what value they are in relation to other questions in that category ($100 for Jeopardy and $200 for Double Jeopardy would have the lowest ordinal score, $500 for Jeopardy and $1000 for Double Jeopardy would have the highest ordinal score). However, the performance was above random guessing but only a 25% improvement with an accuracy on the test set of 25% (random guessing was at 20%).

Michael Bartee [16] also had a study on their GitHub page that is the most like this study except they utilized additional features like the category topic and had an accuracy of 15.6%. They did an additional study just grouping together high versus low values ($1,200 being the high-end boundary). This led to a 67% accuracy but a low recall of 15%.

VI. Conclusion

Given the low accuracy in this model and low performance from the related works mentioned, it comes to question if the valuation of the questions from the head writer is accurate. Questions that are too hard or too easy are already removed from the gameshow making the range of difficulty much narrower than the questions developed by the writing staff.

There are things one can do to update the model that has a possibility of pulling more insights from the dataset. The best but also least likely to be available would be to conduct user research which in this case would be to interview the head writer. They are the primary decision maker on what the values will be for each problem so understanding his decision making would help determine what goes into the categorization.

Changing how the classes are divided would be another solution to understanding the valuation of the problems. Like Awish's analysis, looking at the value compared to the rest of the topic category could reveal some insights and the models listed could have an improvement in performance compared using BERT alone.

Additionally, one could only look at the round the question was placed in or having the topic categories grouped by similar topics with each broader topic having its own models for analysis. Models by topic would make more sense since it would focus the models and have a small vocabulary set which could help with performance (but also lower the number of examples per model).

VII. References

(1) Awards | Jeopardy.com. (n.d.). Www.jeopardy.com. Retrieved December 13, 2022, from https://www.jeopardy.com/about/awards

(2) Grosvenor, C. (2019, March 23). Jeopardy! A Brief History. Live About; Dotdash Meredith. https://www.liveabout.com/jeopardy-past-and-present-history-1396954

(3) Inside the Jeopardy! Writers Room | J!Buzz | Jeopardy.com. (2017, March 17). Www.jeopardy.com. https://www.jeopardy.com/jbuzz/cast-crew/inside-jeopardy-writers-room

(4) J! Archive. (n.d.). J-Archive.com. https://j-archive.com/

(5) Tunguz, B. (n.d.). 200,000+ Jeopardy! Questions. www.kaggle.com. https://www.kaggle.com/datasets/tunguz/200000-jeopardy-questions

(6) Pandas. (2018). Python Data Analysis Library — pandas: Python Data Analysis Library. Pydata.org. https://pandas.pydata.org/

(7) Python. (2009). re — Regular expression operations — Python 3.7.2 documentation. Python.org. https://docs.python.org/3/library/re.html

(8) nltk.corpus package — NLTK 3.5 documentation. (2012, December 12). Www.nltk.org. https://www.nltk.org/api/nltk.corpus.html

(9) tf.keras.preprocessing.text.Tokenizer | TensorFlow Core v2.3.0. (n.d.). TensorFlow. https://www.tensorflow.org/api_docs/python/tf/keras/preprocessing/text/Tokenizer

(10) sklearn.preprocessing.LabelEncoder — scikit-learn 0.22.1 documentation. (2019). Scikit-Learn.org. https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.LabelEncoder.html

(11) Pennington, J. (2014b). GloVe: Global Vectors for Word Representation. Stanford.edu. https://nlp.stanford.edu/projects/glove/

(12) Gensim: topic modelling for humans. (n.d.). Radimrehurek.com. https://radimrehurek.com/gensim/

(13) TensorFlow Hub. (n.d.). Tfhub.dev. Retrieved December 13, 2022, from https://tfhub.dev/google/collections/bert/1?utm_source=www.tensorflow.org&utm_medium=referral

(14) Ghoshal, R. (n.d.). jeopardy-value-prediction. Kaggle.com. Retrieved December 13, 2022, from https://www.kaggle.com/code/rajarshighoshal/jeopardy-value-prediction

(15) Awisk. (2021, April 29). Jeopardy! What, like it's hard?: Using BERT Classification to Estimate Jeopardy Question Valuation. Medium. https://awisk.medium.com/jeopardy-what-like-its-hard-using-bert-classification-to-estimate-jeopardy-question-valuation-488ae115f602

(16) Bartee, M. (2018, April 20). Trivial Pursuit: Jeopardy! Questions and the Value of Knowledge. GitHub. https://github.com/msbartee/predicting-jeopardy-values