Deconstructing Game Design: Communities, Prediction, and Generation for Board Gaming

Nicholas Canu, Kuan Chen, & Rhea Shetty

Project Overview

With access to modern design and publishing tools, board gaming has experienced a historic publishing boom over the past two decades. In the past five years alone revenue for the total market segment grew 170%[1]. While critics ponder if this current boom represents a "golden age" for the industry[2] and the study of ludology in micro for design grows as an area of interest, board gaming continues to have a dearth of domain-scope analysis in the vein of other markets, such as film, that straddle the line between art and commerce. With this in mind, this study intends to deconstruct game data retrieved from BoardGameGeek.com[3] (BGG) and apply multiple machine learning approaches to reveal undocumented patterns in this domain.

Motivations & Objectives

This need for more robust machine learning applications in the field of ludology underpins this study; modern design analysis largely confines itself to discussions on game theory while digital humanities approaches are limited to historically oriented projects such as Ludeme[5]. To bridge gaps between these domain spaces, we will deepen our understanding of game structures by applying ML techniques to a subset of BGG ranked games published from 2000 to 2022. This study continues research from The Impact of Crowdfunding on Board Games (2022)[4] by N. Canu, A. Oldenkamp, and J. Ellis, which provided a proof-of-concept exploratory analysis of BGG as a resource for data in this domain.

To glean meaningful insight on the structure of both individual games and their community as a whole, this study utilizes broad spectrum analysis with multiple foci. This includes unsupervised modeling for topics and document similarity in publisher provided game description text, with supervised prediction tasks for user ratings, game classification based on category, and NLP text feature generation. Leveraging these applications in concert will improve the quality of our analysis as a representative survey of the underlying structure of this BGG dataset.

Ethical Considerations

While the ethical stakes of this analysis remain relatively low, new developments in the relationship between ML & AI solutions and art suggests ways that implementations based upon could become problematic. It has become clear these advances represent a meaningful concern for research in this domain. As tools for generating art systematically become computationally inexpensive and publicly available, debate grows as to their role in relationship to the artist and how they may harm[6] or empower[7].

In intentionally designing applications intended to synthesize with a traditionally artisan-driven design process, we must consider how our tools could be used to remove the human element from game publishing in the future. This will inform their scope and the extent to which they could theoretically replace steps in the design process. In this study we focus on building toward predictive and generative tools that inform how a designer or publisher enhances their games but cannot wholecloth replace their work.

Data Sources

This analysis utilizes a dataset of approximately 22k extracted game items with categorical and quantitative features that create a nearly comprehensive profile of a ranked game on BGG. The data extends from a processed dataset generated by The Impact of Crowdfunding on Board Games (2022)[4] which is, in turn, originally based on stream data scraped via the BGG XML API in April 2022 by @mshepherd on GitLab and retrieved from the Recommend.Games repository.[8] Both are licensed under Creative Commons Attribution-NonCommercial-ShareAlike.

No additional datasets are utilized in this study, unranked game items are excluded, and user & review datasets also which were scraped, were discarded prior to import.. While the final dataset was extracted via a csv file and imported with parquet.gzip due to feature size making other storage types computationally infeasible, the stream files were originally exported in .jl format using the DataStreams.jl library.

As an enthusiast website, BGG represents an imperfect proxy for this domain as a whole but, as the largest forum for board gaming, it does represent a major driver of market focus and might reasonably offer insight into potential structures in games as a whole. Regarding data points selected - to be ranked on BGG a game must have a minimum of thirty user ratings. At this point it will have a bayesian-weighted rating generated through a proprietary calculation that balances average rating against number of votes and the age of the game. We chose to limit the scope of our data as unranked games are, generally, less feature complete. The key supervised prediction targets for this study are the aforementioned user rating, category labels approximating the "genre" of a given, and text features including titles and descriptions. We also looked at predictions for the number of users rating a game as a secondary target.

A major challenge in the format of this data comes from how BGG encodes categorical features; descriptive features, such as category and game type, or publishing features, such as artist, publisher, or designer, are encoded in nested lists. To expand this data to a usable representation, it was transformed via one-hot encoding. After transforming the data our finalized set of key predictors included min/max values for player count, recommended player age, and game complexity alongside categorical features outlined above; not all feature sets are leveraged in all models and a significant portion of our data review consisted of evaluating information gain per feature grouping.

Data Preparation

While the imported dataset had some pre-processing performed including string and value formatting after import we focused on four key manipulations to prepare the data for analysis. We expanded features to re-encode categoricals, controlled project scope by limiting on release year, split feature group subsets and training sets to test different approaches on, and imputed missing values. Select categorical fields had single value strings on initial import, game items with a single game type as an example, but many categorical fields had nested lists with multiple strings represented as a single string value.

We cleaned the remaining fields such as artist, designer, and publisher using cleaner functions licensed from the previous team's analysis[4] then expanded them into one-hot encoding with a new feature column for each value in the dataset. This brought the total feature size to over thirty thousand features leading us to further segment feature subsets to test on. Future implementations might consider the relative value of modeling on the entire dataset but this was computationally prohibitive in our environment and, furthermore, reduced predictive accuracy on specific models as discussed in their respective analyses.

Games as old as 3500BCE[9] exist in the data so choices about project scope were called for. To determine a reasonable threshold, we combined domain knowledge about publishing patterns with an analysis of releases over time. As seen here in figure one, taking releases from 2000CE on represents more than the upper three quantiles of data points while including the elbow for the acceleration, not pictured, that coincides with increased availability of crowdfunding as identified by previous studies.

The Acceleration of Game Releases by Decade - 1950-2020

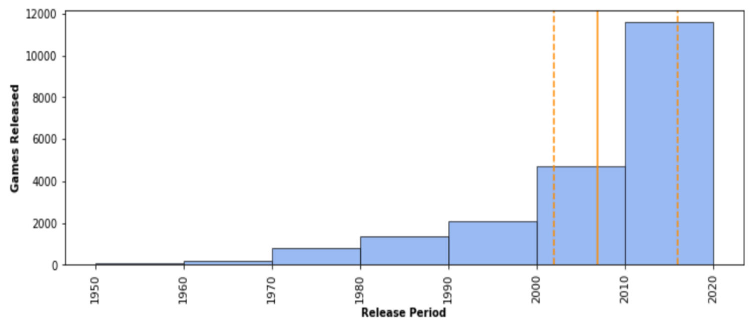

Fig. 1 - A histogram depicting all ranked game items published between 1950 & 2020 binned by decade of release with quartiles delineated

To create feature subsets, we selected prefix tagged one-hot columns in conjunction with a selection of core features included in every set. These features favored publisher provided values to reduce potential data leakage while discarding post hoc user recommended values. Features that could reasonably be provided prior to user interaction, including family tags, were retained despite typically being provided by users in the BGG ecosystem.

Finally we imputed a small subset of missing values with a simple imputer. Unsupervised approaches to imputation were tested and found to provide minimal benefit over an imputer that filled mean feature values due to the sparsity of missing values. One category, cooperative, had a large null count due to an encoding error but we restored the correct values without imputing by confirming that it was intended as a binary flag for tracking cooperative versus competitive base game modes. With these processing steps in place we began examining unsupervised approaches to further generating features and identifying critical features to orient toward.

Decomposition

Given the feature space encoded in this dataset, it became computationally impractical to pass the entire dataframe as input to our predictors. One potential solution to this came through using reduced feature-selected data subsets. However, we also chose to test unsupervised dimensionality reduction to identify the most valuable components in the data to pass to downstream prediction tasks. To perform this reduction, we identified two potential python libraries - sklearn[10] and prince[11].

Initial iteration focused on principal component analysis (PCA)[12] and singular value decomposition (SVD)[13] through sklearn. Before reducing subsets of data and additional preprocessing it required nearly 2000 components to explain 90% of total variance in the data; we identified two major factors impacting the performance of both models. The sparsity of one-hot encoded vectors created a high amount of noise in each decomposition, causing a rapid increase to 60% variance but stalling afterwards, amplified by failure to correctly scale and normalize standard encoding features. A key concern through all iterations was to avoid divesting individual features within groups prior to decomposition; as these represent categorical flags for all possible labels within that original feature space, splitting them would remove possible predictor values for classifier models.

Given mixed performance on these initial tests, we considered if the combination of nominative, ordinal, and continuous data might give more performant decomposition using flexible approaches such as factor of mixed data (FAMD)[11] which can handle both quantitative and categorically encoded data as input. Pursuing this line of research to improve performance we pivoted to the prince library, however it not only performed more slowly but the variance explained by this approach converged more slowly than our initial implementation. Initial analysis of this decomposition failed by overlooking how re-encoding multiple data types to a sparse one-hot matrix caused some factor analysis implementations to no longer treat features as categorical. At the same time, we had leveraged proper preprocessing steps to utilize them as quantitative features.

To resolve this impasse, we used MinMax scaling to bound all features between values of 0 and 1. This normalized the difference in visible variance between large subsets of one-hot encoded features and the smaller group of high information value quantitative features.

Explained Variance by Number of Components in SKLearn SVD

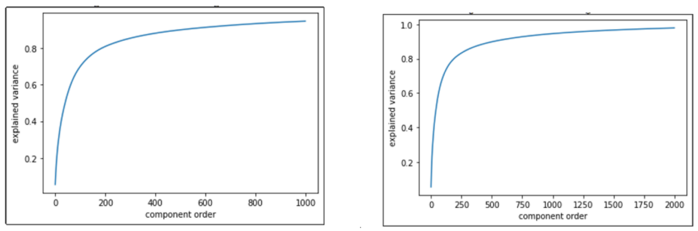

Fig. 2 - Side-by-side plots showing the variance explained per component by SVD with preprocessed data using the SKLearn SVD implementation

The best resulting SVD model, as seen in figure 2, utilized these scaled features with a subset of core features, game type, mechanic, and family to produce 90% variance explained in only 550 components using the sklearn implementation. Other tested subsets of features increased more quickly than our initial attempts once scaled but did converge on the target threshold as quickly, helping identify this grouping as a priority subset for testing feature importance in our supervised models.

Topic Modeling

While this dataset includes informative features, in a real-world setting they may not provide enough information alone to engage a user with a given title. Game descriptions provide a holistic, human-interpretable, summary of the game and bridge the gap for users between mechanical bullet points and a fully envisioned game experience. With over 17,000 games in this dataset, identifying description themes and patterns manually becomes an infeasible task; leveraging topic modeling allows for doing this programmatically.

This unsupervised learning technique automatically clusters data using patterns extracted from a provided text corpus. These patterns may include, as examples, the frequency of phrases or words. Beyond identifying these undiscovered patterns, we sought to generate an imputed feature set to improve supervised model performance. We considered several common unsupervised methods for topic modeling, narrowing our focus to two: Latent Dirichlet Allocation (LDA) and Non-Negative Matrix Factorization (NMF).

LDA functions as a probabilistic model[14], it assumes each document is composed of many words and that some words act as the most representative of a topic. By examining word probability distribution for a document, it sorts the word probability for each topic. It continuously updates topic assignments based on words discovered in different documents and multiplying the probability of a topic given a document by the probability of a word given a topic. LDA has a few notable differentiating characteristics; it assumes more than 1 topic within a document, provides different assignments every time the model runs regardless of the state of the corpus, and excels at extracting longer form documents.

In contrast, NMF implements a linear algebraic model similar to PCA which reduces the dimensions of a given non-negative matrix[15] (A) into 2 matrices (W x H). This provides a good approximation of the original matrix in a lower dimensional space. In this model, A represents a term document matrix of m by n dimensions, W stands for a topic matrix of dimensions m by k, and H contains the weights of topics for dimensions k by n. This lower dimensional space extracts meaningful features to use as topics. For Identifying characteristics of NMF, it does not assume multiple topics per document and uses weights to determine the best-suited topic for a document, produces consistent output across iterations, and more effectively extracts meaningful information with smaller text sequences.

We ran both models to test if these characteristics[16] held true by examining the top ten words of derived topics. Before we performed this topic modeling over our corpus, we also implemented text preprocessing. If we input nonsensical words then the model will use them and likely fail to find meaningful topics.

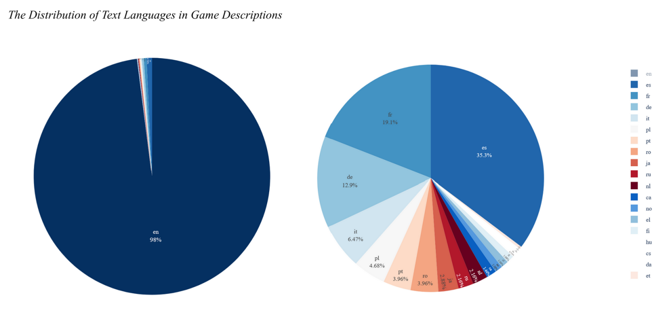

Fig. 3 - Two pie charts depicting the % breakdown of languages present in the game item description corpus with and without English

Initially examining the descriptions raised a concern that the dataset contains multiple languages which might pose an issue in modeling. However, after running a language identification library langdetect, we discovered ~98% of the corpus consists of English text with Spanish at a distant second place as seen in figure 3.

We processed description text using a custom function combining regex replacement with a spAcy[17] NLP tokenizer passed into a configured preprocess string function from the Gensim library; this combination gave us fine-grain control of exactly which processing steps to include. Once we collated a list of lemmatized text, we then removed a set of multilingual stopwords using the NLTK[18] English and Spanish libraries. Domain-specific words to remove were identified through iterations of examining top words in our text corpus. These cleaned up descriptions contain meaningful words to facilitate the model identifying coherent themes.

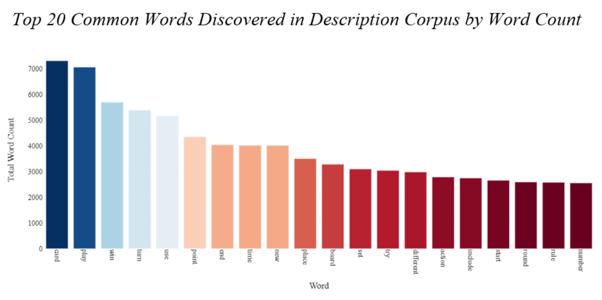

Fig. 4 – A color-graded bar chart showing the total in-corpus word count for the top thirty most common words in the cleaned description text corpus

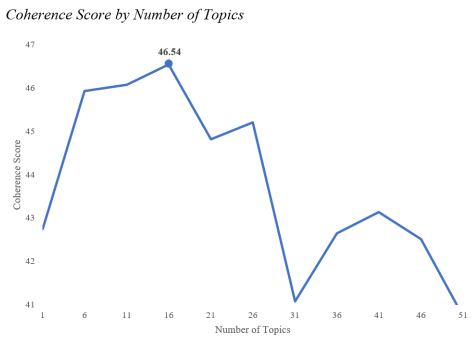

Fig. 5 – A line plot showing the coherence score value at each number of topics to 51

With the text corpus processed, it needed additional conversion to a format interpretable by the model. As a matrix factorization method, NMF input can include any non-negative matrix. To create this new input, we transformed the text corpus with the term frequency-inverse document frequency (TF-IDF) calculation. TF-IDF outputs a matrix representing the statistical proportion of text relative to the corpus as a whole. This provides the model with more robust semantic features to understand the text. LDA, as a probabilistic model, only requires the frequency of words in any given document collection. To prepare for LDA, we simply transform the text into a bag-of-words matrix representation.

We implemented Scikit Learn NMF and LDA[19] models based on ease of use and their comprehensive documentation. As an unsupervised learning technique, topic modeling still needs human input; the only required parameter is a number of topics. This parameter impacts the qualities of output topics which we measure and evaluate by their coherence scores[20]. A higher coherence score means more semantically related words within the topic and is determined by the distance between words within a topic. Closer word vectors have a higher semantic relevance. Over eleven iterations with five additional topics in each iteration, we gathered the coherence scores and discovered sixteen topics performed best with a 46.54 coherence score.

In iteratively testing topic configurations, NMF consistently delivered stronger human-interpretable topics identifiable as different game genres. LDA topic word distributions honed in on the style and function of a game. To transform these topic features, we factorized the TFIDF matrix and used the weights matrix to populate topic features against the original data series. As seen in figure six, particularly topics seven and nine, the topic structure of these descriptions aligns with intuition that the provided text shares some relationship with game genre.

Best Distribution of Words by Topic in an NMF Model

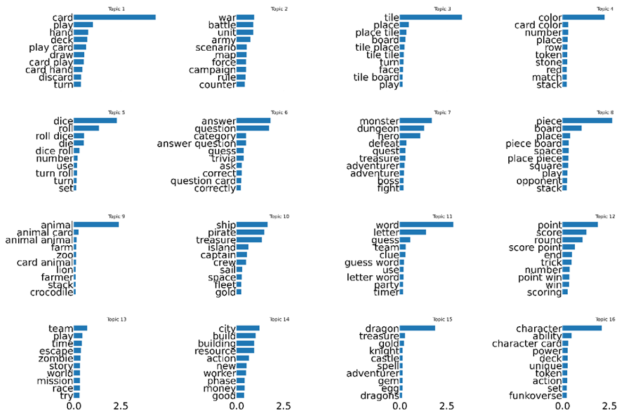

Fig. 6 - Small multiples bar plots depicting the relative distribution of words within each human readable topic in the best performing NMF model.

Document Similarity

As another inflection point for considering potential community structures, our analysis examined a long-standing hypothesis about the categorization of game designs. In the domain a commonly held popular dichotomy exists between opposed communities of design - Euro-style vs American and strategic vs thematic genres, as examples, remain common cross-topic differentiators for games; these have permeated the domain to the degree that Eurogames have their own wikipedia page highlighting how their unique traits versus other subgenres[21]. Do these dynamics manifest in the BGG data?

To explore this question of dichotomous communities, and provide features for downstream tasks, we examined the text similarity of board game descriptions. By decomposing each to build a bag-of-words corpus using the custom text processing function outlined above, we generated a cosine similarity matrix of training game descriptions scored against one another. These descriptions have a somewhat unusual global structure with a few key topics per document and use similar, domain-specific, language but, at the same time, their size varies anywhere from a single sentence to several paragraphs; with this in mind, cosine similarity projection into multi-dimensional space made it a superior choice to other low compute cost measures where varying document length would impact model performance. We used gensim's implementation[22] due to integrated functions for querying new documents against existing matrices via updated indexing supporting our downstream tasks.

The similarity index takes a bag-of-words corpus as input; our text processor developed for topic modeling worked to generate these from the descriptions without modification. However, without standardization, the corpus contained enough noise that nearly all vectors had similarities within a .1 spread. We identified, through customized query functions, domain specific stopwords that mitigated this. After generating the document corpus, the function creates a temporary index file and provides both to gensim.Similarity, which returns an index storing cosine similarity for each document against all others. Iterating through that index extracted a dense similarity matrix. Gensim queries new documents by updating this index with a fresh corpus scored against all extant vectors. We implemented functionality at this stage to return given similarity weights as an imputed feature in a dataframe to test within in our NLP generator.

In analyzing the similarity matrix, an argument for rejecting common belief around design dichotomy did emerge when considering the language by which publishers self-identify their titles. In passing calculated weights as edges between spring-loaded game nodes in two-dimensional space[23], the similarity matrix forms a concentric shape with a single dense inner cluster and sparse outer ring as seen in figure seven. This example shows the top five thousand ranked games in our training set with a similarity threshold of 0.3.

Document Cosine Similarity Vectors as a Weighted Network

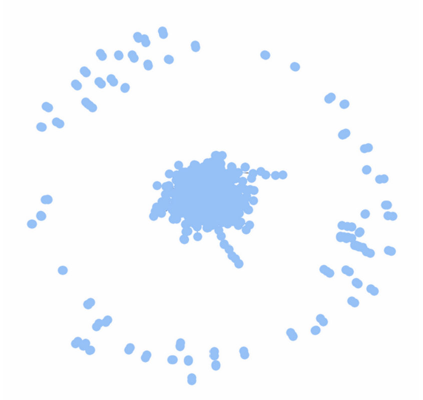

Fig. 7 - A graph depicting cosine similarity of the top 5k ranked games with edges representing similarity scores above 0.3

Even when pruned of common domain words, the language in these descriptions, measured by relative distance, remains consistent. This suggests that, at least for publishers, a common language of descriptors exists with outlying points of less common linguistic choices roughly equally common in how much they diverge. If the anticipated dichotomous pattern had emerged, we would expect two large binary clusters, similar to the central one pictured, indicating that game descriptions did self-select into defined domain roles.

To build for downstream tasks and analyze communal structures in the data, we attempted clustering methods to evaluate potential communities in game descriptions. Ultimately this approach did not contribute additional meaningful imputed features to inform predictive tasks but failed due to identifiable causes that could be rectified by future implementations.. Given the shape of similarity vectors in 2D space, intuition suggests that center and spread based methods would result in poor clustering. Spectral clustering[24] fits directly on pre-constructed affinity matrices but the quantity of data points here created barriers to inference due to increasing compute cost.

Similarity Clusters Identified by N=2 Spectral Clustering

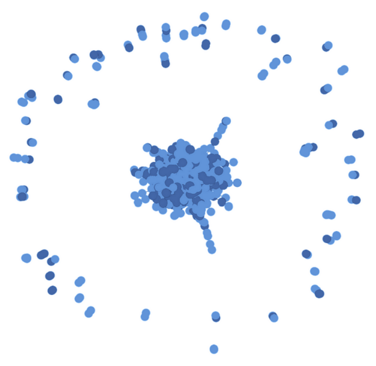

Fig. 8 - The same weighted graph in fig. 7 with labels extracted from spectral clustering encoded by color

Two cluster splits, as seen in figure eight, scored best on both silhouette[25] score and Calinski-Harabasz[26] index relative to other cluster splits. Scores bottomed out at cluster values of two to ten before increasing slightly as cluster counts entered the thousands. This suggests that, given enough splits, spectral clustering could define distinct ring clusters but after iteratively testing fifteen hundred clusters, increasing compute cost made it practically infeasible. These scores both attempt to measure cluster dispersion, and are not as effective on non-convex clusters, the relative separability of outlying clusters in this data suggested they would identify ideal clusters - based on the visualized clustering it is plausible they do not adequately capture the similarity vector space.

While these clustering approaches failing to resolve does not feed forward, it suggests the language used to describe games does not align neatly into bins as the pattern emerging from these descriptions does not sort the games meaningfully. When applied to a dataframe and reviewed, the inter-cluster variance for key features, not used to generate this model, remains low. Based on these findings, we propose that dichotomies espoused in board gaming domain discussions do not extend to the way individual games choose to present themselves and may, further, not reflect true design boundaries; we suggest further investigation in this area to better characterize the common language used by games to present themselves.

Response Prediction

As the key target of this study, bayes_rating represents the underlying value that controls game ranking on BGG. It aggregates and weighs avg_rating, the average of the unweighted ratings given by users, and num_votes which represents the total votes on a given game item. To prevent games with few votes from filling the rankings, and to stabilize ranking fluctuations, BGG computes this Bayesian average by adding "dummy" votes scoring 5.5 to the average rating until the vote count reaches a certain quantity. BGG considers the exact calculation a black-box value and does not publish the input values.For this first supervised task we attempted prediction of ratings based on a group of core features and various one-hot feature groups. Since the target value contains continuous data, we implemented regression models for this task.

As the overall structure of the categorical one-hot data would not change, we chose to develop a demonstration model to identify best approaches for this dataset before finalizing input data features. For this initial run, we collated core features such as min & max age, complexity, min & max players, alongside one-hot feature groups for mechanics, families, categories, and game type. The core numerical features were input without transformation. After data preprocessing we applied a single train test split of 80/20%. As outlined above, imputers did not cause test models to return significantly different results due to the small amount of missing values and density of the data. We implemented the chosen simple imputer and passed the remaining data to each model.

Selected test models included k-nearest neighbor regressors, linear regressors, random forest regressors, and gradient boosting regressors. These ran on default parameters with R² score used as an evaluation metric. As observed in the associated table, most models underfit the data. Random forest showed the greatest potential for improvement with hyperparameter tuning and further feature selection. This aligned with intuition that given the mix of data types and fullness of the data, a random forest that can split at each decision point would more effectively fit the training data.

User Rating Model Selection Performance on Mechanics/Category/Family/Type Development Set

Regression Model	R^2
K-Nearest Neighbors	Training set score: 0.53 Test set score: 0.25
Ridge	Training set score: 0.80 Test set score: 0.60
Random Forest	Training set score: 0.94 Test set score: 0.61
Gradient Boosting	Training set score: 0.62 Test set score: 0.56

In machine learning applications it is equally important to have both an accurate and interpretable model. Feature importance[27] refers to the relative importance of each feature encoded in the training data. This describes which selection of features provide most predictive power on the target variable; it can also help via dimensionality reduction to speed up model implementation

When fitting a random forest regressor, the model provides a *feature_importances_* property to access for retrieving the relative importance scores for each input feature. As an initial test, we took selections of top n features from this output chart and retrained the regressor, evaluated them again using R² scores. Observe in the associated table that 400 features maximized the test set score at 0.63; this score maintained at 500 features but beyond that started trending downwards which indicated that, when selecting features in this subset, we should consider limiting their scope. Despite this, the gross fluctuation observed in scores also indicated that model sensitivity to this selection remained low.

Top N Feature Selection in Random Forest Regressors

Number of top features selected	Random Forest R² scores
300	Training set score: 0.95 Test set score: 0.62
400	Training set score: 0.95 Test set score: 0.63
500	Training set score: 0.95 Test set score: 0.63
1000	Training set score: 0.95 Test set score: 0.62
2000	Training set score: 0.95 Test set score: 0.62

For the development model we performed tuning on the dataset containing the top 400 features identified by feature importance selection on a random forest regressor using GridSearchCV in sklearn. After iterating through this process, cross validation scores for identified peaked at 0.62. This indicated that to improve model performance we needed to evaluate the feature group input.

Having tested model performance with our initial feature combination, we progressed to evaluating feature selection both within a model but also the quality of feature input to the model. Because our feature groups represent one-hot encodings of categorical labels, and reducing features had not significantly improved performance, we decided to avoid feeding partial groupings to future models; to address this task we developed additional evaluation metrics.

Taking our best performing development model, we calculated base information from predictors based on R² score for the training and test sets using the core feature subset. By calculating net information gain[28] at n1 and n2 combinations for all groups alongside the size of each feature space we identified input features valued highest in the tradeoff between compute cost/predictive power balance. We determined that, for bayesian ratings, publisher, mechanics, game type, and family were priority features; the low gain features, in most model configurations, reduced predictive scores when added to high feature models due to additional feature noise outweighing the benefits of additional data points.

Information Gain per Feature Group for Rating Prediction with RandomForestRegressors

Group	Train Gain	Test Gain	Feature Count
Base Scores	0.898	0.302	8
Artist	0.013	0.036	11079
Category	0.007	0.048	84
Designer	0.015	0.049	9998
Family	0.026	0.156	5083
Game Type	0.023	0.139	12
Mechanic	0.027	0.157	183
Publisher	0.047	0.294	6772

Leveraging this improved input selection we immediately saw improvements in untuned RandomForestRegressors with R² test scores of 0.698, up from 0.63, indicating that initial input selection had missed key predictors in the data. By tuning hyperparameters we discovered default sklearn package settings often produced the best results with increasing numbers of trees in each random forest providing incrementally better train and test performance. Settings which reduced the fullness of the trees caused consistent drops in both train and test performance due to the density of input data.

With feature selections updated, we discarded other previously tested models and passed the refined input through a selection of models appropriate to high-dimensional data. While random forest models showed improvements to predictive power, histogram-based gradient boosting regressors (GBR)[29] quickly became evident as the best model for this target and data. While this model continues the additive forward-model of GBR in fitting weak-learner regression trees, it improves performance for large samples by binning values to reduce splitting points and building predictions based on histograms for each bin instead of continuous values of each feature. Avoiding sorting feature values through the histogram structure solved the issue we encountered in iterative testing for standard GBR and justified the compute cost to run models on the fullest viable feature set to maximize performance.

Best Performance per Regression Model on User Rating with Selected Hyperparameters

Regression Model	Best Train R2	Best Test R2	Tuning
Decision Tree	0.612	0.523	max_depth=7
Hist-Gradient Boosting	0.908	0.755	max_iter=350, max_leaf_nodes=30, early_stopping=False
HGBR - Ada Boost	0.936	0.768	n_estimators=15
Random Forest	0.958	0.702	n_estimators=400, max_depth=100
Support Vector	0.864	0.588	epsilon=0.2

Random forest plateaued in performance above four hundred estimator trees, but with a similar number of estimators, Hist-GBR executed in approximately half the time at a 0.05 higher predictive score. A maximum score of 0.755 R^2, with additional features only lowering performance, indicated that this dataset may exclude key confounder variables that impact the predictive accuracy on user rating.

To offset the structure of one-hot categorical features impacting performance by having even weighting across all sub-features in a group, we passed the best performing regressor through an AdaBoostRegressor[30] in the sklearn ensemble module. These meta-estimators take in any other regressor, the indicated Hist-GBR in this case, and iteratively fit while learning new weightings from the error of each predictor. When extended to fifty iterations, a process taking several hours, performance gain plateaued at fifteen iterations suggesting the maximum gain from reweighting features to not be substantial and supporting our intuition that confounding variables are absent in the data.

As a secondary goal, we reimplemented this same analysis pipeline with the number of user votes per game item as the target. In performing feature selection, we discovered that, within this dataset, only the publisher feature group provided substantial information gain above core feature performance. Game type and mechanics added a small amount of information to the predictor with all other groups harming baseline prediction regardless of model selection. In this implementation the best performing model, a random forest with 250 trees, capped its R^2 score at 0.59 on the test set. The divergence between performance on rating and number of votes reinforces the absence of confounders; from a domain-oriented perspective, in consideration of the work done by previous studies, the lack of pre-release financial data including, but not limited to, professional review data, crowdfunding metrics, and marketing presence likely impact predictive power.

Given the importance of publisher features across both regression modeling tasks, an initial analysis of the data suggests that *who* releases a game may play a greater role than *what* design features that game contains in driving user preferences. We propose that further analysis of BGG data, if the aforementioned data cannot be integrated, uses instrumental variables to control for these unknown confounders; this approach might produce stronger evidence of direct causal relationships between specific game features and rating predictions while helping to potentially characterize, more concretely, information gaps in the data.

The target values studied here change daily. In conjunction they control overall ranking on BGG which represents a constantly shifting hierarchy with data outdated as soon as it's frozen in a snapshot. Generalizing these predictions on complex aggregate metrics in a live environment will require more testing and iteration on historical and live data than possible in the scope of this study. However, we have identified several approaches to reasonably estimate user ratings while offering paths forward for developing additional predictors and improving collective understanding of data relationships in this domain.

Game Classification

The genre of a game, or the category in BGG parlance, inform a user's perception of a game even prior to playing it; at one step in the descriptive hierarchy below game type, this feature helps users select new games that align with their interest. Being able to classify game categories using mechanical, family, and game types categorical features alongside imputed description features could provide a tool to better align game items to genre in the future. In this case a game may possess more than one category label, making this a multi-label classification task and guiding our approach to it. In case of multi-label classification, each data point can simultaneously belong to one or more classes of target variables and predicted classes are not mutually exclusive. This approach thus assigns each sample a set of target labels.

As this task would use text features, it required additional processing including cleaning the description column as described in our unsupervised models to the point of lemmatizing the text. We passed these lemmatized descriptions as a transformed feature with the remaining features in their one-hot encoded format. In this model, we filled missing data in target variables with a 'no category' string and encoded the remaining target variables with sklearn's MultiLabelBinarizer[31]. Before extracting features from the cleaned descriptions, we implemented an 80/20% train test split. We experimented with contiguous sequences of n-grams, unigrams, bigrams and trigrams, for creating tf-idf vectors and observed that bigram models yielded the best score with 10,000 max_features and min_df = 0.01 as parameters limiting the total scope of n-gram features.

To perform multi label classification we transformed the output into a set of binary classification problems using the One vs Rest (OvR) classification strategy. This consists of fitting one classifier per identified class and, for each classifier, fitting it against all other classes. It learns a binary model for each class that attempts to separate the class, resulting in binary models per class equal to the total count of classes - 85 for this data. While testing we classify the sample as belonging to the maximum scoring class. In this process we trained four models including logistic regression, gradient boosting, linear SVC and multinomial NB; of these linear SVC produced the best f1 score. This model allows for easy control of regularization parameters and trains relatively quickly given that it assumes a linear kernel while still allowing for a OvR implementation.

Multilabel Classifier Predictive Performance

Model Name	F1 Score	Hamming Loss	Recall	Precision
Logistic Regression	0.581	0.0206	0.4509	0.8177
Gradient Boosting	0.551	0.023	0.4466	0.7196
LinearSVC	0.609	0.0216	0.53	0.7147
Multinomial NB	0.511	0.0253	0.4177	0.6592

In multi label classification[32], prediction of an instance can be fully correct, partially correct or fully incorrect. Because of this, accuracy does not align with the goals of model evaluation. In this instance we scored micro averaged F1 as the key performance indicator. In addition to this F1 score we also tracked hamming loss, recall, and precision as secondary KPIs. Micro averaged F1[33] fit the problem well because it weights based on the frequency of a label occuring and works well with a highly imbalanced distribution. Hamming loss also provides valuable insight as it evaluates, on average, the frequency that a label is incorrectly assigned. Dummy classifiers and PR curves were used to evaluate model efficiency. These individual scores per model appear in the associated table. However, scores alone do not provide a complete picture of these evaluation metrics when controlled by the regularization hyperparameter C in linear SVC. Performing sensitivity analysis on this hyperparameter aids interpretation of model variability and generalizability.

F1 Scores by Regularization Parameter C

C	Train Accuracy	Test Accuracy
0.0001	0.111	0.112
0.001	0.25	0.244
0.01	0.464	0.432
0.1	0.731	0.589
1	0.942	0.609
10	0.993	0.552
100	0.997	0.53
1000	0.995	0.527
10000	0.991	0.525

F1 Scores Measured at Levels of C

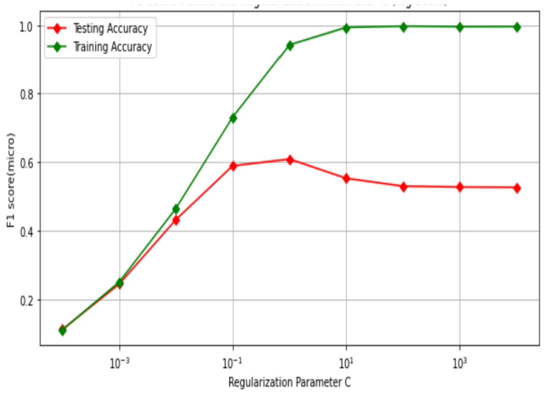

Fig. 9 - A line chart depicting the divergence of training and test set accuracy measured by F1 score at multiple levels of regularization.

Increases in C link to increases in F1 scores until C reaches a value of one. Larger values represent less regularization which causes the model to fit the training set with as few errors as possible. From a C of ten and up, the model begins overfitting to the training data. After performing 10-fold cross validation and hyperparameter tuning, the model achieves a maximum micro-averaged F1 score of 0.60, recall value of 0.519, precision value of 0.758 and hamming loss of 0.021 using LinearSVC. We can manually observe categories assigned to each game using MultiBinarizer's inverse function.

While this model shows promise in evaluating and predicting the BGG categories linked to game genre, more work must occur to fully classify game items automatically. With the availability of varied classification strategies, including other classification strategies within these same models, we foresee potential improvements coming from further evaluation of how the data structure interacts with varied classification strategies beyond OvR which does not appear to ideally fit the data.

Text Generation

To date we have analyzed potential community structures in BGG data and attempted to predict response variables based on the game items provided; however, automatic feature generation remains the final goal for machine learning applications on this data. In this study we will provide a proof-of-concept for its utility by generating and evaluating game titles based on input descriptions. As a complex task with many parameters to understand and establish natural language rules, we have chosen to perform text generation via a neural network (NN) model[34]. We utilized the Great Lakes Clusters to leverage their large computing resources necessary for NN model development.

Text generation takes a trained natural language model and generates new, ideally relevant, phrases. A title represents a sequentially related text string, not a random assortment of words. As an example, the title of this project, Deconstructing Game Design, forms a sequence of words related to one another in ways that cannot be disentangled without losing the meaning of it. In this case, focus on the key sentence structures via its composing words; due to the part of speech property, a verb, of the first word, the second word will likely be a noun or adverb to accommodate it. We can examine the word relevance of a sentence because the first word's meaning directly causes the second word to represent something that can be acted upon derived from the first word. As this language model will learn grammar and semantics without any initial framework, we observe the importance of how a rich, diverse, and large text corpus can impact model performance.

As a subset of all NLP tasks, the preprocessing tasks outlined above are executed with modifications tailored to the generator. Excluding stop word removal protects words integral to a title that the model should learn to decode sentence structures. Additionally, this step now converts titles into n-gram sequences. With the project title as an example, the output appears in this structure - [“deconstructing game”, “deconstructing game design”]. Finally padding sequences helps vectors align into a matrix[35].

Also we consider the sequential nature of words in a sentence by the probability of the presence of each word following the first[36]. This generator learns probabilities from a sequence of words to predict the next word as a target (y) and treat everything before as predictors (X) likened to a classification task. Recurrent NN (RNN)[37] has shown effectiveness in modeling tasks because it retains memory of past data; however RNN also faces the issue of vanishing and exploding gradients caused by repeated multiplication of next word probabilities. Long Short-Term Memory Cell (LSTM) architecture mitigates this by turning multiplication into addition or subtraction thus retaining the information for long sequences of text and outputting more stable results.

With many parameters[38] available to a neural network[39] we focused on tuning[40] four. Two model complexity parameters include number of units per layer and number of layers which regulate how detailed and well a model learns parameters from the data including grammar and sentence structures: Two other parameters, learning and dropout rate, target regularizing the model to perform better on previously unseen data to control generalizability.

The number of units or width of a model determines how detailed a structure the model learns from data. Given enough width, it essentially memorizes training data, not a desired result for a text generator constructing from already-produced titles. The number of layers or depth of a model teaches it to think more abstractly about the task but too deep layers result in a model that can only return the most basic output; for example, exclusively generating stop words since they represent the most frequent in a corpus. The learning rate controls how quickly a model adapts to the data via adjusting weights to minimize prediction loss. Setting this parameter too high causes it to learn bad weights that produce subpar output while with a too low rate it may never learn the weights. Dropout rate[41] controls a random percentage of node output to be set to 0 limiting the learned parameters. This should minimize codependency among nodes from prior training and improve generalizability.

All four parameters require balance to achieve a model that can train efficiently yet performs the task at hand accurately. This model additionally minimizes the categorical crossentropy[41] loss function for a multi-class classification task. This improves text generator performance in learning to classify the most appropriate next words from the model dictionary in a one-hot encoded data format. We experimented with parameter configuration and, as the learning rate decreased and dropout increased, observed validation loss sharply curving down after a handful of iterations alongside training loss plateaus. We found the adam optimizer’s default learning rate coupled with a 20% dropout rate produces consistent improvement in training loss while maintaining gradual validation loss.

BLEU Scoring for Generated Text by Modifier

Modifier	**Generated**	**Reference**	**BLEU**
Clipped Precision	the the the the	the game of thrones	1/4=0.25
Order Precision	thrones of game the	the game of thrones	0/4=0

To evaluate this task, we used two metrics, Bilingual Evaluation Understudy (BLEU)[42] and Gensim word2vec similarity score[43]. These should evaluate how closely the generator produces new text with high semantic and structural similarities given the test data seed text. BLEU measures the structural similarities of text by generating a score comparing a given text sequence to its reference text by matching the new sequence n-gram of the new sequence to that of the reference sequence. It employs modifiers to account for repeated words and word order. It also uses clipped precision aimed to discount single repeated matches. For word ordering precision, it calculates multiple n-grams of the reference and generated text for matches (2-grams, 3-grams, 4-grams, etc) then averages them. Gensim's word2vec similarity measures the cosine similarity between given text. The word2vec model maps text vectors to an embedding space where closer vectors are more semantically. This functions as a good comprehensive measure of how well the model generates meaningful phrases.

Text Generator Model Performance Metrics

Model#	Nodes	hidden layers	Learning rate	Dropout	Min Loss	Min Val Loss	AVG BLEU	AVG Similarity
Model1	256	2	adam default	0.2	3.3	10.6	0.096335	-0.003428
Model2	256	2	exponential decay	0.4	7.5	9.8	0.09361	-0.003428
Model3	256	4	adam default	0.2	7.3	8.8	0.044347	-0.001809
Model4	512	4	adam default	0.2	4.7	9.3	0.09467	-0.003428

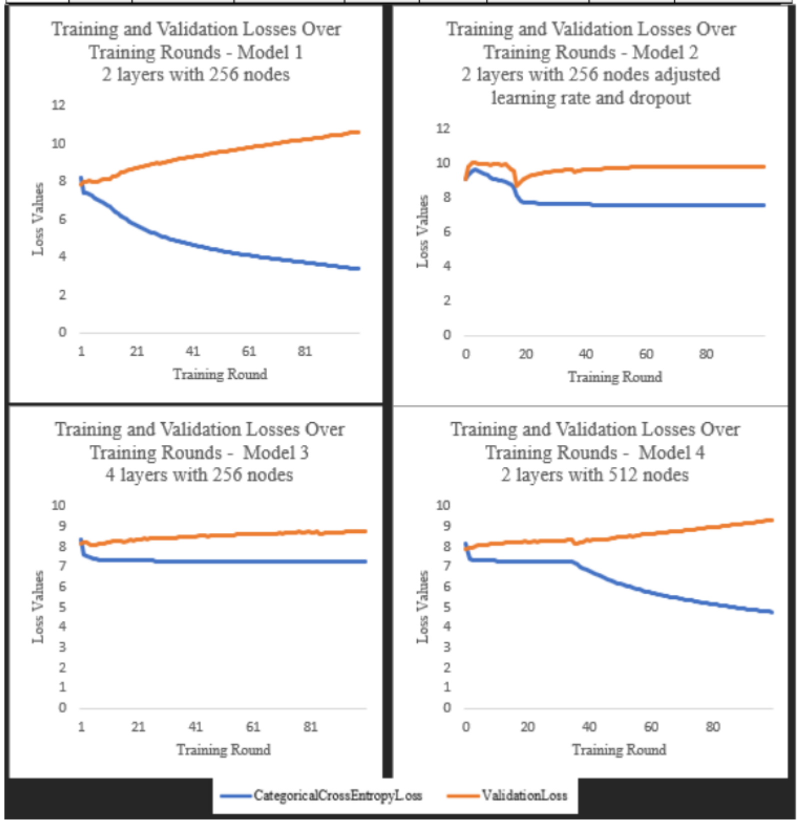

Fig. 10 - A table and small multiples line plot showing model best model performance and performance over training rounds

Among four tested models, model 1 performed best across all metrics but it overfits. Model 2 shows less overfitting but could not produce coherent title names. While model 4 performs slightly worse than 1 in these metrics, it overfits less and produces reasonable, human-readable, game titles. In our modeling, two hidden layers with 512 nodes performed best on training loss but produced higher validation loss, indicating overfitting despite adjustment to regularization.

This can be attributed to the small amount of text data available for training as most neural networks require millions of data points. Another area of improvement[44] includes instead having the model learn at the character level to provide granular information on the sentence structures and reconfigure the architecture[45] of the network[46].

Conclusions & Reflections

Throughout this study each task sought to extend understanding of hidden structures within game data and empower downstream models. By identifying topical and document similarity patterns, we fostered insight into how games self-identify through interpretation of their descriptive text. While this dataset did not provide the full scope of features needed to maximize predictive power, it performed well at predicting and interpreting user ratings, and less so for the number of votes and categorical labeling. Through the lens of feature group selection, we identified priority features for each predictor and suggested potential future extensions of the data. And in revealing these patterns and engaging with model development, we created a proof-of-concept demonstration for one application on this dataset by building a title generator.

Several fundamental challenges arose across all tasks. With a high-dimensional feature space, the compute cost for modeling stressed available resources. Dimensionality reductions could not provide significant enough gains to justify a relative loss of predictive power in tested models. Considering the data structures, fullness of models coincided directly with changes in performance and data reductions impacted prediction in most tasks. Future work should implement distributed computing solutions from the beginning to enable efficient iteration over models; this issue limited our ability to test at scale and our implementation of unsupervised models such as clustering. The need for informative feature groups also emerged as a recurrent theme. Though the best performing models effectively utilized available data, others did not achieve high predictive performance with any feature set; this indicates significant, but as yet unknown, confounding variables existing outside this dataset. To advance understanding of how to best represent BGG as a dataset, identifying and integrating potential features will bridge the gaps found in this study.

From the outset this analysis acknowledged the dearth of previous work in the domain and sought to build upon what did exist; our approach took these disparate tasks and synthesized a holistic picture of how features relate within available data as well as how to model upon that knowledge. Future studies should seek to enhance the completeness of available data, extend developed models to fully map identified communities and discover more granular relationships, and explore more robust generative models from the frameworks developed here. We hope this study empowers analysts with baseline knowledge of board gaming as a domain and a richer understanding of how to engage with BGG specifically as a data source.

Work Summary

All team members contributed to project proposals and reports. Individual contributions to specific tasks are as follows:

N. Canu provided collaboration workspaces, acquired and pre-processed data, worked on text processing, analyzed document similarity, developed the stage two regression models, generated the report framework and non-task specific drafts, and edited the final report.

K. Chen explored decomposition strategies, led development of text processing solutions, performed description topic modeling and feature imputation, worked on stage one regression and classification models, and built the title generator neural network.

R. Shetty analyzed feature correlation and importance, created the framework for all supervised predictors, led development of stage one regression and all classification models, and performed hyperparameter tuning for supervised models.

References

1. "Board Games - Worldwide." Accessed October 17, 2022. https://www-statista-com.proxy.lib.umich.edu/outlook/dmo/app/games/board-games/worldwide
2. “Board Games' Golden Age: Sociable, Brilliant and Driven by the Internet.” The Guardian. Guardian News and Media, November 25, 2014. https://www.theguardian.com/technology/2014/nov/25/board-games-internet-playstation-xbox.
3. “BoardGameGeek.” Accessed October 17, 2022. https://boardgamegeek.com/.
4. Canu, Nicholas, Jonathan Ellis, Adam Oldenkamp. *The Impact of Crowdfunding on Board Games*. Jupyter Notebook, 2022. https://github.com/canunj/BGG_KS_Analysis.
5. “Digital Ludeme Project.” Accessed October 17, 2022. http://ludeme.eu/.
6. Roose, Kevin. “An A.I.-Generated Picture Won an Art Prize. Artists Aren’t Happy.” *The New York Times*, September 2, 2022, sec. Technology. https://www.nytimes.com/2022/09/02/technology/ai-artificial-intelligence-artists.html.
7. “AI Image Generators Will Help Artists, Not Replace Them | BBC Science Focus Magazine.” Accessed October 17, 2022. https://www.sciencefocus.com/news/ai-image-generators-will-help-artists-not-replace-them/.
8. Shepherd, Markus. “Recommend.Games / Board Game Scraper · GitLab.” GitLab. Accessed October 17, 2022. https://gitlab.com/recommend.games/board-game-scraper.
9. BoardGameGeek. “Senet.” Accessed October 17, 2022. https://boardgamegeek.com/boardgame/2399/senet.
10. Pedregosa, Fabian, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, et al. “Scikit-Learn: Machine Learning in Python.” *Journal of Machine Learning Research* 12, no. 85 (2011): 2825–30.
11. Halford, Max. *MaxHalford/Prince*. Python, 2022. https://github.com/MaxHalford/prince.
12. “In Depth: Principal Component Analysis | Python Data Science Handbook.” Accessed October 22, 2022. https://jakevdp.github.io/PythonDataScienceHandbook/05.09-principal-component-analysis.html.
13. Halko, Nathan, Per-Gunnar Martinsson, and Joel A. Tropp. “Finding Structure with Randomness: Probabilistic Algorithms for Constructing Approximate Matrix Decompositions.” arXiv, December 14, 2010. http://arxiv.org/abs/0909.4061.
14. Egger, Roman, and Joanne Yu. “A Topic Modeling Comparison Between LDA, NMF, Top2Vec, and BERTopic to Demystify Twitter Posts.” *Frontiers in Sociology* 7 (2022).
15. Coyler, Adrian. “The Why and How of Nonnegative Matrix Factorization.” Blog. the morning paper, n.d. https://blog.acolyer.org/2019/02/18/the-why-and-how-of-nonnegative-matrix-factorization/.Egger, Roman, and Joanne Yu. “A Topic Modeling Comparison Between LDA, NMF, Top2Vec, and BERTopic to Demystify Twitter Posts.” *Frontiers in Sociology* 7 (2022). https://www.frontiersin.org/articles/10.3389/fsoc.2022.886498.
16. spaCy 101: Everything you need to know. “SpaCy 101: Everything You Need to Know · SpaCy Usage Documentation.” Accessed October 20, 2022. https://spacy.io/usage/spacy-101/
17. “NLTK :: Natural Language Toolkit.” Accessed October 20, 2022. https://www.nltk.org/

18. scikit-learn. “Topic Extraction with Non-Negative Matrix Factorization and Latent Dirichlet Allocation.” Accessed October 21, 2022. https://scikit-learn/stable/auto_examples/applications/plot_topics_extraction_with_nmf_lda.html.
19. Kapadia, Shashank. “Evaluate Topic Models: Latent Dirichlet Allocation (LDA).” Medium, December 29, 2020. https://towardsdatascience.com/evaluate-topic-model-in-python-latent-dirichlet-allocation-lda-7d57484bb5d0.
20. Zvornicanin, Enes. “When Coherence Score Is Good or Bad in Topic Modeling?” Blog. Beeldung, December 7, 2021. https://www.baeldung.com/cs/topic-modeling-coherence-score.
21. “Eurogame.” In *Wikipedia*, August 15, 2022. https://en.wikipedia.org/w/index.php?title=Eurogame&oldid=1104458448.
22. “Gensim: Topic Modelling for Humans.” Accessed October 17, 2022. https://radimrehurek.com/gensim/similarities/docsim.html.
23. Hagberg, Aric A., Daniel A. Schult, and Pieter J. Swart. “Exploring Network Structure, Dynamics, and Function Using NetworkX.” In *Proceedings of the 7th Python in Science Conference*, edited by Gaël Varoquaux, Travis Vaught, and Jarrod Millman, 11–15. Pasadena, CA USA, 2008.
24. Luxburg, Ulrike von. “A Tutorial on Spectral Clustering.” arXiv, November 1, 2007. http://arxiv.org/abs/0711.0189.
25. Shutaywi, Meshal, and Nezamoddin N. Kachouie. “Silhouette Analysis for Performance Evaluation in Machine Learning with Applications to Clustering.” *Entropy* 23, no. 6 (June 16, 2021): 759. https://doi.org/10.3390/e23060759.
26. Maulik, U., and S. Bandyopadhyay. “Performance Evaluation of Some Clustering Algorithms and Validity Indices.” *IEEE Transactions on Pattern Analysis and Machine Intelligence* 24, no. 12 (December 2002): 1650–54. https://doi.org/10.1109/TPAMI.2002.1114856.
27. Jason Brownlee. "How to Calculate Feature Importance With Python" March,2020. https://machinelearningmastery.com/calculate-feature-importance-with-python/
28. Larose, Daniel T., and Chantal D. Larose. *Discovering Knowledge in Data: An Introduction to Data Mining*. Second edition. Hoboken: Wiley, 2014.
29. scikit-learn. “Sklearn.Ensemble.HistGradientBoostingRegressor.” Accessed October 22, 2022. https://scikit-learn/stable/modules/generated/sklearn.ensemble.HistGradientBoostingRegressor.html.
30. scikit-learn. “Sklearn.Ensemble.AdaBoostRegressor.” Accessed October 22, 2022. https://scikit-learn/stable/modules/generated/sklearn.ensemble.AdaBoostRegressor.html.
31. scikit-learn. “Sklearn.Preprocessing.MultiLabelBinarizer.” Accessed October 21, 2022. https://scikit-learn/stable/modules/generated/sklearn.preprocessing.MultiLabelBinarizer.html.
32. MMA. “Metrics for Multilabel Classification.” Mustafa Murat ARAT, January 25, 2020. https://mmuratarat.github.io//2020-01-25/multilabel_classification_metrics.

33. Paul, Saugata. “A Detailed Case Study on Multi-Label Classification with Machine Learning Algorithms And….” Medium (blog), June 17, 2019. https://medium.com/@saugata.paul1010/a-detailed-case-study-on-multi-label-classification-with-machine-learning-algorithms-and-72031742c9aa.
34. Amazon Science. “How to Make Neural Language Models Practical for Speech Recognition,” August 29, 2019. https://www.amazon.science/blog/how-to-make-neural-language-models-practical-for-speech-recognition.
35. KDnuggets. “Tokenization and Text Data Preparation with TensorFlow & Keras.” Accessed October 21, 2022. https://www.kdnuggets.com/tokenization-and-text-data-preparation-with-tensorflow-keras.html.
36. “6 The Universal Workflow of Machine Learning · Deep Learning with R, Second Edition.” Accessed October 21, 2022. https://livebook.manning.com/deep-learning-with-r-second-edition/chapter-6.
37. Karakaya, Murat. “Char Level Text Generation with an LSTM Model.” Deep Learning Tutorials with Keras (blog), March 19, 2021. https://medium.com/deep-learning-with-keras/char-level-text-generation-with-an-lstm-model-e55ba7ff18c2.
38. Team, Keras. “Keras Documentation: The Functional API.” Accessed October 20, 2022. https://keras.io/guides/functional_api/.
39. “Tips for Training Recurrent Neural Networks.” Accessed October 20, 2022. https://danijar.com/tips-for-training-recurrent-neural-networks/.
40. Brownlee, Jason. “A Gentle Introduction to Dropout for Regularizing Deep Neural Networks.” Article. Machine Learning Mastery, n.d. https://machinelearningmastery.com/dropout-for-regularizing-deep-neural-networks/.
41. 1. Peltarion. “Categorical Cross Entropy Loss Function | Peltarion Platform.” Accessed October 20, 2022. https://peltarion.com/knowledge-center/modeling-view/build-an-ai-model/loss-functions/categorical-crossentropy.
42. Brownlee, Jason. “A Gentle Introduction to Calculating the BLEU Score for Text in Python.” Blog. Machine Learning Mastery, n.d. https://machinelearningmastery.com/calculate-bleu-score-for-text-python/.
43. Li, Zhi. “A Beginner’s Guide to Word Embedding with Gensim Word2Vec Model.” Medium, June 1, 2019. https://towardsdatascience.com/a-beginners-guide-to-word-embedding-with-gensim-word2vec-model-5970fa56cc92.
44. Farzad, Amir, Hoda Mashayekhi, and Hamid Hassanpour. “A Comparative Performance Analysis of Different Activation Functions in LSTM Networks for Classification.” Neural Computing and Applications 31, no. 7 (July 1, 2019): 2507–21. https://doi.org/10.1007/s00521-017-3210-6.
45. Kingma, Diederik P., and Jimmy Ba. “Adam: A Method for Stochastic Optimization.” In 3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May

7-9, 2015, Conference Track Proceedings, edited by Yoshua Bengio and Yann LeCun, 2015. http://arxiv.org/abs/1412.6980.

46. Zhuang, Zhenxun, Mingrui Liu, Ashok Cutkosky, and Francesco Orabona. “Understanding AdamW through Proximal Methods and Scale-Freeness.” ArXiv, 2022.