Topic modeling and categorization of board games using LDA and TF-IDF

Your Name

April 7, 2025

Abstract

This paper presents a textual analysis approach applied to comments related to various board games. The methodology involves the following steps: data preprocessing, topic modeling using Latent Dirichlet Allocation (LDA), and topic categorization based on TF-IDF vectorization and cosine similarity. We also present visualization strategies to support the interpretation of the results.

1 Introduction

Topic modeling is a fundamental task in the field of Natural Language Processing and with this project we used the boardgames comments, collected from `https://boardgamegeek.com/browse/boardgame` to derive topics for each of the first ten games in the leaderboard. In this work, we propose an approach that involves data preprocessing, topic modeling, and finally the analysis of the derived topics in order to assign boardgames to predefined categories.

2 Theoretical Background

2.1 Latent Dirichlet Allocation

Latent Dirichlet Allocation (LDA) is a generative probabilistic model for collections of discrete data such as text corpora. It assumes that each document is a mixture of a small number of topics and that each word in the document is attributable to one of the document's topics.

Mathematically, LDA defines:

- A Dirichlet prior distribution over topic distributions for each document.
- A Dirichlet prior distribution over word distributions for each topic.

The model is trained using algorithms such as Gibbs Sampling or Variational Bayes, and it returns for each document a distribution over topics, and for each topic, a distribution over words.

2.2 TF-IDF: Term Frequency-Inverse Document Frequency

TF-IDF is a statistical measure used to evaluate the importance of a word in a document relative to a corpus. It combines two metrics:

- **Term Frequency (TF)**: Measures how frequently a term appears in a document.
- **Inverse Document Frequency (IDF)**: Measures how important a term is, i.e., it gives less weight to common terms.

The formula is generally given by:

$$\text{TF-IDF}(t, d) = \text{TF}(t, d) \cdot \log\left(\frac{N}{df(t)}\right)$$

where N is the total number of documents and $df(t)$ is the number of documents containing term t.

2.3 Cosine Similarity

Cosine similarity is a metric used to measure the similarity between two non-zero vectors in a multi-dimensional space. In text mining, it is often used to compare TF-IDF vectors of documents.

The cosine of the angle between two vectors $\vec{A}$ and $\vec{B}$ is given by:

$$\cos(\theta) = \frac{\vec{A} \cdot \vec{B}}{\|\vec{A}\|\|\vec{B}\|}$$

The result ranges from 0 (completely dissimilar) to 1 (identical in direction).

3 Data Collection and problems:

In order to obtain documents, we used the XML api of Bgg site, with a small script in python we achieved this goal. We decided to separate this part of code from the main project because this "collecting comments" code takes a huge amount of time. Firstly the analysis was thought to involve more boardgames, but downloading comments we realized that a lot of them were "no sense": a huge number of non-English comments, but also empty comments or singular letters without any meaning. So we decided to focus on the first ten boardgames, downloading the maximum possible number of comments for them, then cleaning them, keeping only the ones in English. So from ten thousand comments per game, we passed to less than the half of the comments, but in some case even less.

4 Data Preprocessing

The data is loaded from a CSV file containing these fields `id`, `names`, and `comments` for each game. The `get_game_comments` function filters comments for a specific game. Text cleaning and tokenization are done using `clean_text` and `preprocess` functions to:

- Remove emails and URLs.
- Remove punctuation and special characters.
- Convert to lowercase.
- Remove stopwords.

Lemmatization is performed by the function lemmatize, using `spacy`, limited to tokens with POS tags `NOUN`, `ADJ`, and `VERB`.

5 Topic Modeling with LDA

After preprocessing, the documents are transformed into bigrams using `gensim.models.Phrases`. A dictionary and corpus are created and passed to an LDA model using `gensim`'s `LdaModel`. The `train_lda` function trains the model, and coherence scores are calculated using `CoherenceModel` to assess topic quality.

```
def train_lda(corpus, dict, ntopics, random_state):
    ldamodel = LdaModel(corpus=corpus, id2word=dict, num_topics=ntopics,
    random_state=random_state)
    return ldamodel

def compute_coherence(lda_model, texts, dictionary):
    coherence_model = CoherenceModel(model=lda_model, texts=texts, dictionary=
    dictionary, coherence='c_v')
    return coherence_model.get_coherence()
```

Listing 1: LDA training and coherence evaluation

For what concerns the hyperparameters, we decided to try different options in order to see which one could be the best option. In this project we adopted a number of topics to search of five.

6 Topic Categorization

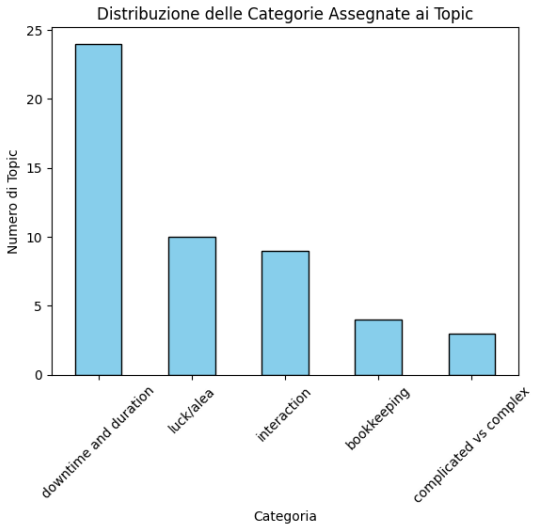
Each topic is matched to a predefined category. Categories include: *luck/alea*, *bookkeeping*, *downtime and duration*, *interaction*, *bash the leader*, and *complicated vs complex*.

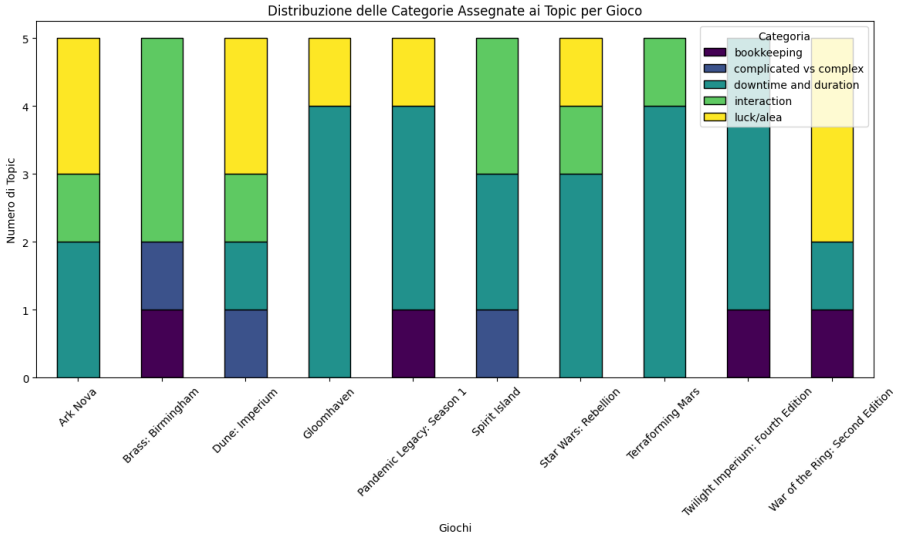
The categorization process involves computing TF-IDF vectors for both the topics and category keywords. Cosine similarity is then used to assign the most semantically similar category to each topic.

```
vectorizer = TfidfVectorizer()
tfidf_matrix = vectorizer.fit_transform(all_texts)

category_vectors = tfidf_matrix[:len(categories)]
topic_vectors = tfidf_matrix[len(categories):]
similarities = cosine_similarity(topic_vectors, category_vectors)

df_results = pd.DataFrame(similarities, index=topic_labels, columns=categories.
    keys())
df_results["Assigned Category"] = df_results.idxmax(axis=1)
```

Listing 2: TF-IDF and cosine similarity for categorization

7 Visualization of Results





Here the repository with the pyLDAvis representation of each game, with their respective topics in order to see which are the most relevant and frequent terms per topic.

`https://github.com/Barbleiss/Knowledge-and-information-retrivial/tree/main/lda_visualizations`

8 Conclusions

In the end, we discovered a lot of interesting topics for each of the ten games we analyzed. Thanks to the pyLDAvis representations, we can identify the most relevant and frequent words and see which terms compose the various topics. For this project, I would like to expand the study to include more games, not just the top ten games on the leaderboard. Additionally, I aim to build an application that can automatically fetch comments from the web in a reasonable time and display these kinds of visualizations and metrics for any game of interest, without splitting the code into two separate stacks. In terms of precision, the approach doesn't show remarkable performance, as we can see that the coherence scores are not very high—they are rather average. What we could try in the future is incorporating trigrams or n-grams in general, applying other preprocessing techniques, and using a different model.