Stardew Summarization:
Unsupervised Document Summarization of a Video Game Wiki System

Alex Wills
Computer Science with Writing
New College of Florida Sarasota
alexander.wills23@ncf.edu

Nisanur Genc
Computer Science/Mathematics
New College of Florida Sarasota
nisanur.genc19@ncf.edu

Abstract

For our final project for Natural Language Processing, we have worked on our research project: topic-specific extractive summarization on the Stardew Valley Wiki. The goal was to analyze the model for summarizing Wikipedia-styled webpages. Using the Stardew Valley Wiki articles as our dataset, where each page is a separate document, we applied an unsupervised summarization model based off of the TextRank algorithm. Then we evaluated the model and analyzed the effectiveness using the ROUGE method (Lin, 2004). Although our algorithm did not perform optimally on average, we suggest some improvements and directions for future research in this area.

1 Introduction

Video game wiki systems, such as the Stardew Valley Wiki (2022), are frequently used by players of the game to find useful information. These wikis are developed by a large open source community of people who play the game, and as a result they are dense with detailed content on a large quantity of topics relating to the game. As such, wikis can be overwhelming for readers. To further complicate the issue, the open source nature of wiki systems leads to articles that do not all follow the same form and may be difficult to navigate.

Our study seeks to explore solutions to the issue of information overload in game wikis by applying an automated summarization algorithm to generate short snippets of wiki pages that a user can use to quickly judge whether they want to read an article. If successful, this algorithm could be implemented into wiki systems to improve the user's experience. Our primary approach for automated summarization of wiki articles follows the TextRank algorithm for extracting the three most important sentences in an article to create a summary.

1.1 Motivation

Although fan-created wikis, like the Stardew Valley Wiki, are organized and contain features to make navigation easier, with over one thousand pages and with pages created inconsistently by many users, these wikis can be intimidating or off-putting to read through due to information overload. This project would benefit people who use game wikis by attempting to summarize pages, allowing for an automated system to compact the wiki's information into a more readable form. Accurate page summaries would also work in the context of Sateli and Witte's (2012) architecture, assisting users in searching for information that is relevant to their needs.

Research has been done into the integration of NLP systems into wiki systems and shows promise for the practical use of such systems (Sateli and Witte, 2012; Witte and Gitzinger, 2007). Our research focuses on the specific problem of text summarization for this area.

To conduct this study, we used an unsupervised model for document summarization. For this model, we used the TextRank algorithm described by Mihalcea and Tarau (2004). We originally intended to compare this model to a supervised LSTM model, but, due to limitations, that model will be left for future research.

2 Related Works

Extractive Summarization is a well-known natural language processing (NLP) task that has extensive research, such as that done by

Mihalcea and Tarau (2004) with the TextRank algorithm. NLP tasks have also been applied to wiki systems by Witte and Gitzinger (2007) and Sateli and Witte (2012), including automated page summarization. Our work focuses on using the TextRank approach in the wiki system context for automated summarization.

The foundation for this project is Mihalcea and Tarau's (2004) TextRank algorithm, which follows Google's PageRank algorithm by using graphs to represent the data. This unsupervised algorithm represents a document as a graph of sentences connected by their similarity to each other. Using "global information recursively drawn from the entire graph," namely the sentences' similarities to each other and their current score for importance, this algorithm repeatedly updates the graph until the rankings of the sentences converge to stable values. This algorithm has been shown to succeed in keyphrase extraction and sentence extraction for summarization, and due to it not requiring specifically-annotated corpora, it is "highly portable to other domains, genres, or languages" (Mihalcea and Tarau, 2004).

Initially, we intended on implementing a neural network to compare to TextRank, modeled after the LSTM model introduced by Hochreiter and Schmidhuber (1997). Since the length of documents in our dataset varies greatly, a recurrent neural network like LSTM would be beneficial for processing documents of varying size. LSTM processes input sequentially, using results from the previous input to inform its decision for the current input. This model could be used to process sentences in a document sequentially to generate a summary.

For evaluating summarization models, researchers commonly use the Recall-Oriented Understudy for Gisting Evaluation (ROUGE) package specified by Lin (2004). Mihalcea and Tarau (2004) used this approach, which uses n-grams to evaluate a model's summary against a gold standard summary, to evaluate TextRank for sentence extraction. Our research uses the same approach, specifically with bigrams.

Witte and Gitzinger (2007) investigated the specific domain of wiki systems and how NLP tasks can be integrated for practical use. They note that one practical application of NLP is in automatic summarization of wiki pages for helping users in "identifying relevant information" and avoiding information overload, although they do not specify a model to use for summarization (2007).

Sateli and Witte (2012) developed an architecture for integrating scalable NLP systems into wiki systems without requiring domain-specific knowledge. In regard to automatic summarization, they suggest the usefulness of generic and focused summaries to assist users navigate multiple pages. Our research explores the generation of generic summaries, which could then be integrated into an architecture like the one designed by Sateli and Witte (2012).

3 Dataset Analysis

We used BeautifulSoup HTML parser in the Python language to create our dataset. We created our corpus by extracting html files as .txt files and processing them with BeautifulSoup. In our dataset we had 1400 files in total.

Each document represents an article on the wiki about the information of some specific subject in the game.

While the maximum number of sentences in a document with the title "Version History" was 1782, the minimum number of sentences in a document with the title "Ore" was 0. The average number of sentences in documents in total was 29. The distribution of the number of sentences in each document is provided in Figure 1.

Although "Version History" is an outlier in its length since it is consistently added to, long documents are expected because some items and characters have more relevant information to list on the wiki than others. Articles on Stardew Valley's characters that have many lines of dialogue, for example, have a considerably high sentence count. "Ore" is the only page with no sentences, and this is because the page only contains links to documents on more specific types of ore. The shorter documents in our dataset typically describe objects in the game that are not very important and do not have much to write about, or are pages that only contain tables of information, which do not count as multiple sentences.

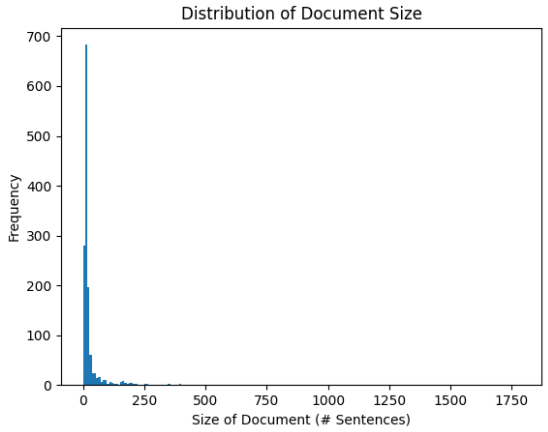

Figure 1: The distribution of the lengths of the documents in the Stardew Valley Wiki dataset.

3.1 Retrieving the Dataset

To run our program, two datasets are required: a corpus, and word embeddings.

We used the word embeddings provided by Pennington et al. (2014), specifically the Wikipedia 2014 + Gigaword 5 pre-trained word vector set. This text file should be found at "data/glove.6B.50d.txt" in the same directory as the TextRank.py program.

Our corpus is a folder titled "Corpus" containing text files representing each page in the game wiki. The corpus comes from the Stardew Valley Wiki (2022), which can be downloaded by running our CreateCorpus.py program. With this program, it is important to ensure that a "Corpus" folder exists in the same directory as the program. This program can be used to create corpora from other wiki systems and websites with multiple pages by modifying the rootSite variable and the extensionsToVisit list in the main function.

4 Methodology Analysis

We conducted this research by extracting articles from the Stardew Valley Wiki website as .txt files. The main reason we chose to create our own dataset was to explore the edge cases in creating a dataset and to learn to handle them correctly. Furthermore, creating our own dataset would allow us to explore the specific area of wiki systems for video games. We used BeautifulSoup HTML parser in the Python language to extract the data, generating 1400 entries for our corpus.

We then wrote a python script to create a Test Set and Training Set from our corpus for future work in comparing our results with a supervised LSTM neural network. This script separates the documents randomly into the Test Set and Training Set, with 10% of the files going to the Test Set and 90% of the files going to the Training Set.

After creating the dataset we preprocessed the data, focusing on tokenization to separate documents into sentences and sentences into words for vector embeddings and ROUGE evaluation. We case folded the sentences and removed the punctuation and special characters from the documents. Due to the nature of text in wiki systems, we removed the header information from the first sentence of every document with regular expressions. We also removed the last two sentences, which contained footer information and page navigation, for every document.

4.1 Unsupervised TextRank Algorithm

The TextRank algorithm determines the most important sentences of a document, which are concatenated to form a summary, by representing the document as a graph of connected sentences with importance scores and repeatedly updating the sentences' scores until they converge and stop changing by significant amounts.

We have implemented the TextRank algorithm by first building a sentence dictionary. This dictionary assigned IDs for each sentence per document and added an initial score for each sentence. The initial score was split evenly between the sentences in the document so that they add to 1. Throughout the TextRank algorithm, this dictionary is updated to adjust scores until they converge.

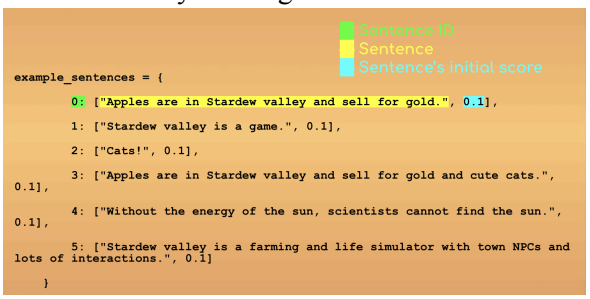

Figure 2: Example sentence dictionary.

The next step was to create a similarity matrix (Figure 3) by calculating the similarity between two sentences with the cosine similarity between the sentence's average word embeddings. These similarity scores represent the edges between sentences in a densely connected graph, whose vertices are the sentences in the sentence dictionary. With indices corresponding to the sentence IDs in the sentence dictionary, one can

quickly look up the similarity between two sentences by indexing the sentence IDs. One can also easily iterate over all of a sentence's connections by only indexing one sentence's row in the matrix and iterating over the row.

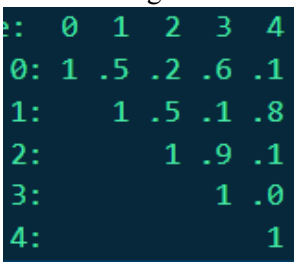

Figure 3: Example similarity matrix. (The bottom left half of the matrix is empty for readability, but in implementation it directly mirrors the top right half).

To update a sentence's score in an iteration of the algorithm, we implemented Mihalcea and Tarau's (2004) formula for considering a sentence's neighbors based on the weight of the edge connecting them relative to the neighbor's score and other edges. A strong connection to a neighbor with a high score and few other strong connections will raise a sentence's score more than a connection with a weak sentence that has many other connections. This formula is applied to every sentence in a single iteration, and it iterates repeatedly until the sentence scores converge.

After the sentences' scores converge to their own values, we extract the sentences with the highest three scores and put them together to form a three-sentence summary.

5 Result Analysis

To evaluate the summaries generated by our implementation of the TextRank algorithm, we used the ROUGE-2 model specified by Lin (2004). For the gold standard baseline summaries to compare with the summaries output by our model, we extracted the first three sentences from each document. Wiki systems are structured in such a way where the first section of every page is a brief overview of the page's topic, so we assume that the first three sentences are relatively strong as a summary of the page.

On average, our model produced summaries with a ROUGE-2 measure of about 0.359. Our highest score was about 0.979, and our lowest score was 0.000. Figure 4 suggests that there may be an association between the length of the document and the quality of the summary produced by the TextRank algorithm.

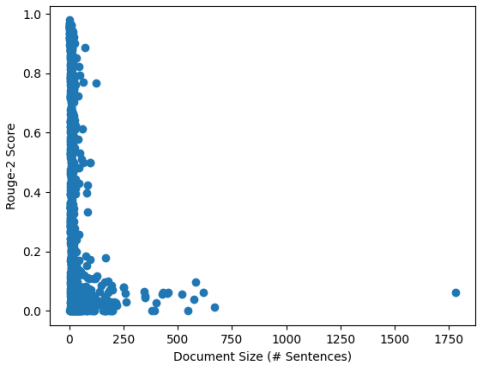

Figure 4: ROUGE-2 scores plotted against document size.

These results suggest that our model may not be very effective in its current state, although we believe that a future version of the model could be adjusted to perform better, rivaling the quality of the TextRank algorithm in other scenarios.

5.1 Input

The input to this algorithm is a text file containing multiple sentences to extract a summary from. The algorithm then splits the document into individual sentences, runs the TextRank algorithm, and generates an output.

The program we wrote takes input in the form of text files in a "Corpus" folder located in the same directory as the Python script.

5.2 Output

At the end of the TextRank algorithm, the three highest-scoring sentences are combined into a single string to output as a summary. Our algorithm also extracts the first three sentences from the document to store as a baseline summary, and it uses ROUGE-2 to evaluate the quality of the found summary.

As a whole, our program outputs a CSV file in the same directory as the program with columns for the file name, the baseline summary, the found summary, and the ROUGE-2 measure.

6 Discussion and Future Direction

Although our model did not perform as well as we predicted, this study still holds potential for further research.

6.1 Improving TextRank

Currently there are a few factors that are possibly related to our model's poor average performance. Our threshold for converging sentence's scores was a net change of 0.001 and our damping

factor was 0.85. It is possible that adjusting these parameters may lead to more accurate results.

Video game wikis, like the Stardew Valley Wiki, commonly present information with long tables. Our model's preprocessing treats entire tables as single sentences with many newline characters, which could distort the graph used in the TextRank algorithm. In a future iteration, we could either ignore tables or process them differently than regular sentences to get a more robust graph for running the TextRank algorithm.

6.2 Evaluation

Further research could explore other means of evaluation for potentially yielding better results. Mihalcea and Tarau (2004) evaluated their model with ROUGE-1. Our study only uses ROUGE-2, but ROUGE-1 or ROUGE-LCS may be a more relevant measurement.

Additionally, our baseline of using the first three sentences as the gold standard summary may not be the most accurate gold standard. Some documents have less than three sentences in their overview section, and sometimes the overview does not comprise the first sentences of the page. Some pages, such as the page for the character Abigail, begin with a quote from the game before the overview. Ideally, the gold standard summaries would be created by human annotators.

6.3 LSTM Neural Network

We would like to investigate the effectiveness of different approaches to automated text summarization in the area of game wikis, as TextRank may not be best suited for pages in a wiki system.

We propose an LSTM model that takes in a text document as input and outputs the three most important sentences. Every sentence could first be assigned a random ID and given either an average word embedding, or a sequence of word embeddings of a fixed length. These sentences could then sequentially be input into the LSTM, which could be trained to output the ID of the most important sentences for the summary.

6.4 Focused Summaries

Satali and Witte (2012) suggest that focused summaries may be more useful for users than general summaries in a wiki system. Users often search for information with a search bar, so the wiki system knows what focuses the user is interested in finding information about.

To modify TextRank for a focused summary, more weight could be given to key words contained in the user's focus, instead of relying solely on word embeddings. Supervised models could also be explored for generating focused summaries.

General summaries may still be useful for users who seek to gain a larger variety of information on a topic in the game wiki, as opposed to users who are looking for specific information.

References

Hochreiter, Sepp and Jürgen Schmidhuber. 1997. Long Short-term Memory. *Neural Computation* 9(8):1735–1780.

Lin, Chin-Yew. 2004. ROUGE: A Package for Automatic Evaluation of Summaries. In *Text Summarization Branches Out*, pages 74–81, Barcelona, Spain. Association for Computational Linguistics.

Pennington, Jeffrey, Richard Socher, and Christopher Manning. 2014. GloVe: Global Vectors for Word Representation. Stanford, CA, USA. Stanford University.

Mihalcea, Rada and Paul Tarau. 2004. TextRank: Bringing Order into Text. In *Proceedings of the 2004 Conference on Empirical Methods in Natural Language Processing*, pages 404–411, Barcelona, Spain. Association for Computational Linguistics.

Sateli, Bahar and René Witte. 2012. Natural processing for MediaWiki: The semantic assistants approach. In *Proceedings of the Eighth Annual International Symposium on WIkis and Open Collaboration (WIkiSym '12)*, Article 10, pages 1-10, New York, NY, USA. Association for Computing Machinery.

Stardew Valley Wiki. 2022. stardewvalleywiki.com/Stardew_Valley_Wiki.

Witte, René and Thomas Gitzinger. 2007. Connecting wikis and natural language processing systems. In *Proceedings of the 2007 International Symposium on WIkis (WikiSym '07)*, pages 165–176, New York, NY, USA. Association for Computing Machinery.