Implementation report for paper of "Story Quality as a Matter of Perception: Using Word Embeddings to Estimate Cognitive Interest."

Mohamed Abdul, Srijit Bhattacharya, Catherine Parmar
California State Polytechnic University, Pomona.
mabdulrahman@cpp.edu, srijitb@cpp.edu, cparmar@cpp.edu

Abstract

Story telling has been an important part of human civilization. In the era of the digital world and technological advancement, story telling has taken a different forms. Modern computers can generate such stories autonomously. In order to have computers generate stories from a human perspective, the generated story needs to be evaluated in an efficient way which can also improve the quality of the generated story by adding specificity, and clarity The paper proposes a technique to evaluate the generated story by analyzing the predictive interference, therefore we have implemented a technique using python programming language, and Bert word embedding library from Hugging face API, and other helper libraries.

Introduction

Many civilization have story telling embedded into their cultures and traditions that tends to hold ancestral significance and a sense of identity. This norm is passed down the successive generation which ensures that the cultures and traditions never die. Story telling have been a powerful tool that tends to shape human lives by giving the meaning of life, preserving traditions, explaining history, entertaining an audience, and so forth. Machines with the correct application can generate stories very much like a human can with the power of imagination. Thus, the aspect of story telling can be improved to generate better stories in terms of the choice of words, creation of vivid imagination with those choice of words, and the quality of language which can enhance the overall human interaction by giving hints for the predictive outcome.

Since machines can generate stories at a much faster rate than humans, a lot of stories can be generated within a short time frame. The idea is to also avoid monotonous stories, and to increase reader engagement. The generated story thus needs to be evaluated. Relying on human subjected to evaluate a story tends to be expensive and time consuming. This paper proposes a mechanism which will evaluate the story using word embedded vectors to assess the predictive inference within the generated story, thus providing a more efficient and an automated evaluation mechanism. The automated mechanism tends to assess how well a story anticipates or predicts furture events and outcomes. Therefore, these are the good indicators of how well the story is generated and how effective it will be in reader engagement.

Procedure

The paper proposes the following procedures to evaluate the stories:

- Eliminate commonly used stop words such as "is", "the","and", etc.
- Eliminate named entities such as people, organizations or locations.
- Remaining words should be used to produce word vectors by utilizing BERT embeddings.
- Compute the running cosine similarity, identify outliers and M score as discussed in the paper.
- Visualize the cosine similarity values against their corresponding words.

Implementation Details

In order to implement the procedure efficiently and more readable in terms of code, we mapped each step of the procedure to a component which does specifically a particular corresponding task. The following utility classes were created to perform some of the required tasks, and some of the wrapper classes of third libraries. The classes were named according to their functionality. With concerns of functionality, some of the utility classes that was created to facilitate the process are was followed:

- **SentenceUtil.py** : Responsible for breaking the story into sentences.
- **EntityRemoverUtil.py** : Responsible for removing named entities such as person, organizations, and locations from text. A third party library

called **spacy** was used to perform the elimination of named entities in a text.

- **StopWordUtil.py** : Responsible for removing stop words from a text. It was not sure what were the list of stop word that original authors have used, hence an arbitrary list of stop words were chosen to perform the task.
- **OutlierUtil.py** : Responsible for finding outliers in running cosine similarity list.

Here are the classes where the majority work is being done.

- **BertProcessor.py** : Responsible for calculating word vectors for words. This is a wrapper class that uses BERT embedding library functions from Hugging Face API. This will produce a list of map *(bertComputedSentences)* containing the word vector values as depicted in the figure 1.
- **StoryQualityEvaluator.py :** Responsible for evaluating the story, and this is the main wrapper class which all of the process happens such as invoking BERT process, calculating cosine similarity, visualizing.

Sample Story

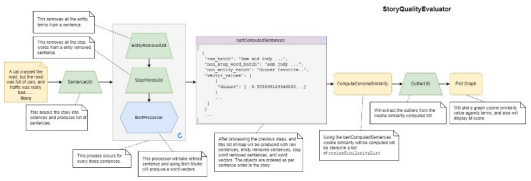

Figure 1: Process of story evaluating to produce visualization of cosine similarity.

The following is a story with 50 words, and the visualization running cosine similarity: "Amelia explored an abandoned mansion, discovering a warning in an old book about a treacherous mirror. Ignoring it, she gazed into the mirror, unknowingly releasing a malevolent entity. Her reflection grew wicked, pushing her toward darkness. Now, she fights to reclaim her true self and overcome the unleashed evil."

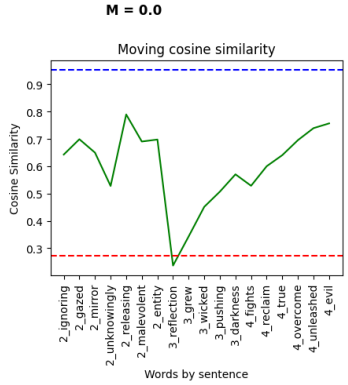

Figure 2: Visualization of story's running cosine similarities

Figure 2 Depicts the moving cosine similarity based on the 50 words abstract which analyzes the word "reflection" as a lower cosine similarity which indicates it might have a higher predictive inference to the story.

How to run our implementation

The development has been done using JetBrains Intellij community version, and also experimented in Visual Studio code. The code can be ran any IDE that user is comfortable with. Under the projects folder, there is **README.md** file, it has list of dependencies to be installed prior to running the program. Once, the dependencies have been installed, and refer the **Experiment.py** which already has demonstration of the API usage, and the user just has to replace story with their story.

Project implementation source code is hosted on the Github: https://github.com/yousuf1997/CS-5990-Term-Project

Summary

The authors did not provide what BERT training model, and stories (besides the example one) have used to conduct their experiment. The team has chosen arbitrary stories to conduct their experiment. The results were convincing that implementation is producing the results as discussed in the paper, but the team did not have a way to validate the results by comparing the results quantitatively. Instead, the team has used their own intuition by reading the story and comparing the graph.

References

Behrooz, M., Robertson, J., & Jhala, A. (2019). Story Quality as a Matter of Perception: Using Word Embeddings to Estimate Cognitive Interest. Proceedings of the AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment, 15(1), 3-9. https://doi.org/10.1609/aiide.v15i1.5217