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 University, Pomona.
mabdulrahman@cpp.edu, srijitb@cpp.edu, cparmar@cpp.edu

Abstract

Storytelling has been an important part of human civilization, in the era of digital technological advancement, the story telling has taken a different form. Modern computers can generate stories autonomously. In order to make computers to generate stories like humans, the generated stories should be evaluated in an efficient way, and hence the quality of the generated stories can be improved. The paper proposed a technique to evaluate the generated stories by analyzing the predictive interference, and we have implemented the technique using python programming language, Bert word embedding library from Hugging face API and other helper libraries.

Introduction

Many cultures and civilizations have been telling stories and passed down the stories to their future generations. Storytelling has been a powerful tool to shape human lives. In the era of digital age, the story telling has taken another form. Machines can generate their own stories autonomously, since the machine have been increasingly interactive with humans in a natural and more like human to human way, the aspect of storytelling by machine can be improved to generate better stories improve overall machine interaction experience.

Since the machines can generate stories in a faster way, and lot of stories can be generated within shorter period of time. In order to avoid monotonous story, and to increase engagement of the reader. The generated stories must be evaluated. Relying on human subjects to evaluate the stories can be expensive and time consuming. The paper proposes a mechanism to evaluate the stories using word embedding vectors to assess the predictive inference within the generated stories, providing a more efficient and automated evaluation mechanism. Basically, it provides an automated mechanism to assess how well a story anticipates or predicts future events or outcomes. These are the good indicators of how well the story is generated.

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 way in code. We mapped each step of the procedure to a component which does specifically a particular corresponding task. So, we have created following utilities classes to perform some of the required tasks, and some of the wrapper classes of third libraries. We named their class according to their functionality. Speaking of functionality, lets begin with some of the utility classes we created to facilitate the process. Here is the list of utility classes we created and their corresponding tasks:

- **SentenceUtil.py** : Responsible for breaking the story into sentences.
- **EntityRemoverUtil.py** : Responsible for removing named entities such as person, organizations, and locations from text. We have used a third party library called **spacy** to perform the elimination of named entities in a text.
- **StopWordUtil.py** : Responsible for removing stop words from a text. We are not sure what are the list of stop word that original authors have used, but we have chosen arbitrary list of stop words 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.

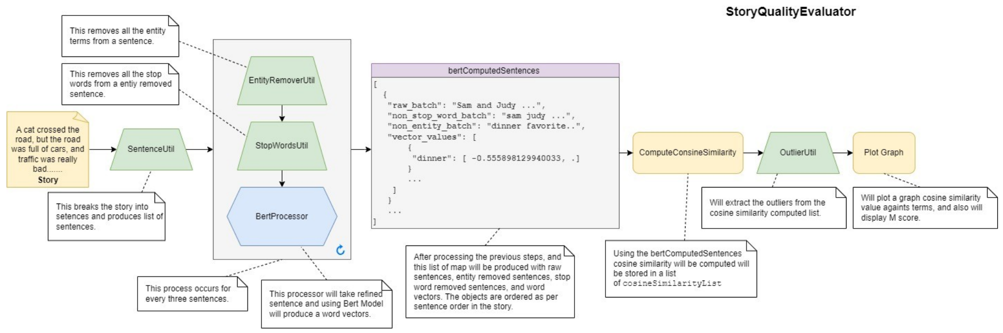
- **BertProcessor.py** : Responsible for calculating word vectors for words. This is a wrapper class that uses BERT embedding library functions from Hugging Face API. This will produce a list of map *(bertComputedSentences)* containing the word vector values as depicted in the figure 1.
- **StoryQualityEvaluator.py :** 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.

Listing 1: Running the API in Experiment.py

```
eval = StoryQualityEvaluator()
eval.initiateBertProcess(story)
eval.computeMovingCosineSimilarity()
eval.plotCosineSimilaritiesBySentenceWord()
```

How to Run

Please refer the Experiment.py in the source file, and listing 1 has a sample core code to run the API.



StoryQualityEvaluator
This removes all the entity terms from a sentence.
This removes all the stop words from a entiy removed sentence.
A cat crossed the road, but the road was full of cars, and traffic was really bad.......
Story
SentenceUtil
This breaks the story into setences and produces list of sentences.
EntityRemoverUtil
StopWordsUtil
BertProcessor
This process occurs for every three sentences.
This processor will take refined sentence and using Bert Model will produce a word vectors.
bertComputedSentences
After processing the previous steps, and this list of map will be produced with raw sentences, entity removed sentences, stop word removed sentences, and word vectors. The objects are ordered as per sentence order in the story.
ComputeConsineSimilarity
Using the bertComputedSentences cosine similarity will be computed will be stored in a list of cosineSimilarityList
OutlierUtil
Will extract the outliers from the cosine similarity computed list.
Plot Graph
Will plot a graph cosine similarity value againts terms, and also will display M score.

```
[
  {
   "raw_batch": "Sam and Judy ...",
   "non_stop_word_batch": "sam judy ...",
   "non_entity_batch": "dinner favorite..",
   "vector_values": [
       {
        "dinner": [ -0.555898129940033, .]
       }
       ...
    ]
  }
  ...
]
```

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

Sample Story

The following is 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."

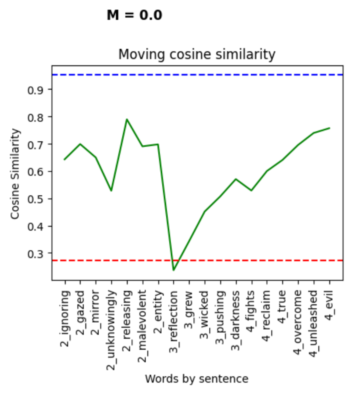


Figure 2: Visualization of story's running cosine similarities