CS584 – Project Milestone

Title:
Text Summarization of Movie Reviews Using Graph Based Ranking Algorithm

Authors:

- Karthik Shivaram
- Nikhil Birur
- Zinuo Li

Problem Overview:
Automatic Text Summarization involves condensing a document or a document set to produce a human comprehensible summary.
They are two approaches to this problem:
a) Extractive Summarization:
In this approach we select a subset of existing words, phrases, or sentences in the original text to form the summary.
b) Abstractive Summarization :
In this approach we build an internal semantic representation and then use natural language generation techniques to create a summary that is closer to what a human might generate. Such a summary might contain words not explicitly present in the original text.

Here our main goal would be to summarize movie reviews created by movie critics so an end user could get a clear picture of how the movie is and if it is worth watching.

Data:
The data we are going to be using here are movie reviews we have collected from rotten tomatoes. Rotten tomatoes is an online movie review database that contains reviews done by movie critics as well as audiences so it is a good resource for the data we need.

We are scrapping rotten tomatoes according to a list of movies using a web scrapper written by us. The scrapper utilizes *urllib* and *beautiful soup* for html parsing so we can get the relevant information we need, i.e. the textual content of the page.

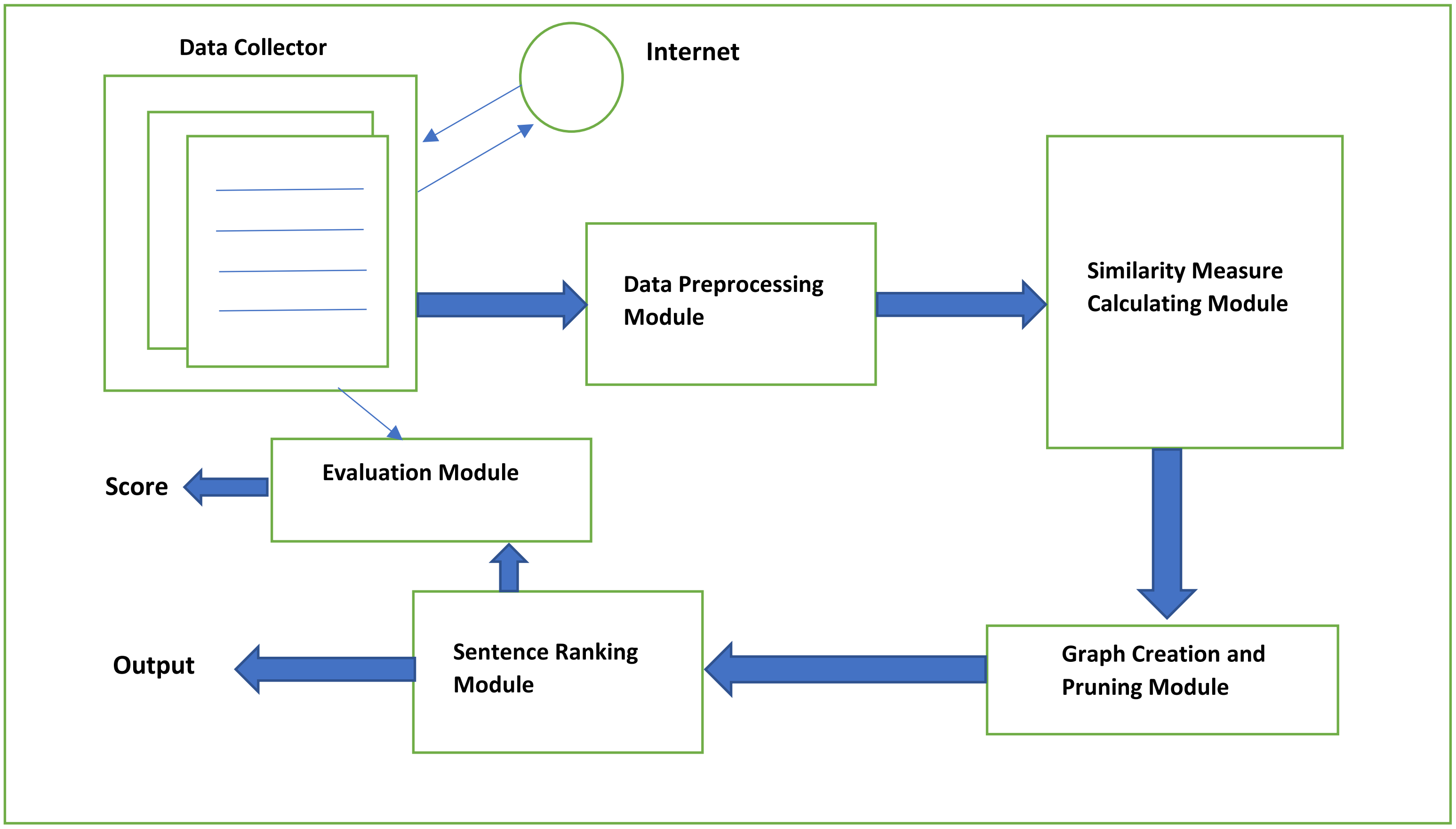
Method:
So here we would be using a modified version of PageRank to rank each sentence in our summary to get the most descriptive one. Even though we are just using our system with data from the movie reviews domain, data from other domains can also be used just as easily.

In our Algorithm after data collection occurs we first identify sentences using a sentence detector and retrieve all the sentences present in our text document after which we send it to the similarity measuring module where every pair of sentences are taken in and the similarity between them is measured.
For the measurement of similarity, we will be using the following scoring metrics:

a) Jaccard Coefficients measure
b) Idf-Cosine Similarity measure
c) Cosine Similarity measure
d) Word Overlap measure
e) Idf-Overlap Similarity measure
f) Phrasal Overlap measure
g) Word2vec Cosine Similarity measure

We are also looking for more similarity measures to capture the information on how similar 2 sentences are with respect to the words use and the context in which they are used.

After the similarity measures are calculated between each sentence pair we send our matrix of scores to our graph creation module which converts this scoring matrix to a weighted undirected graph. Once the graph creation is done we check to see how many edges are present that have a value greater than a ***threshold*** we define, and we prune the edges that are not above this threshold. This step decreases our graph density. Then we will be applying a modified version of PageRank (to take into consideration edge weights) to rank our sentences based on graph centrality (thereby scoring the most important sentences with a higher score). Then we return the sentence or sentences with the highest score.


Data Collector
Internet
Data Preprocessing Module
Similarity Measure Calculating Module
Evaluation Module
Score
Graph Creation and Pruning Module
Sentence Ranking Module
Output

Libraries we are using:

1) Networkx
2) Sklearn
3) Urllib
4) beautifulSoup
5) matplotlib

We are not modifying any part of these libraries. All the steps and algorithms mentioned above are being developed by us from scratch, we are not using any built-in library functions.

<u>Evaluation Methods:</u>

We are using ROUGE (Recall-Oriented Understudy for Gisting Evaluation) based evaluation metrics, they are as follows:
1) ROUGE -N: Measures unigram, bigram, trigram and higher order n-gram overlap

2) ROUGE-L: Measures longest matching sequence of words using LCS (Longest common Subsequence)

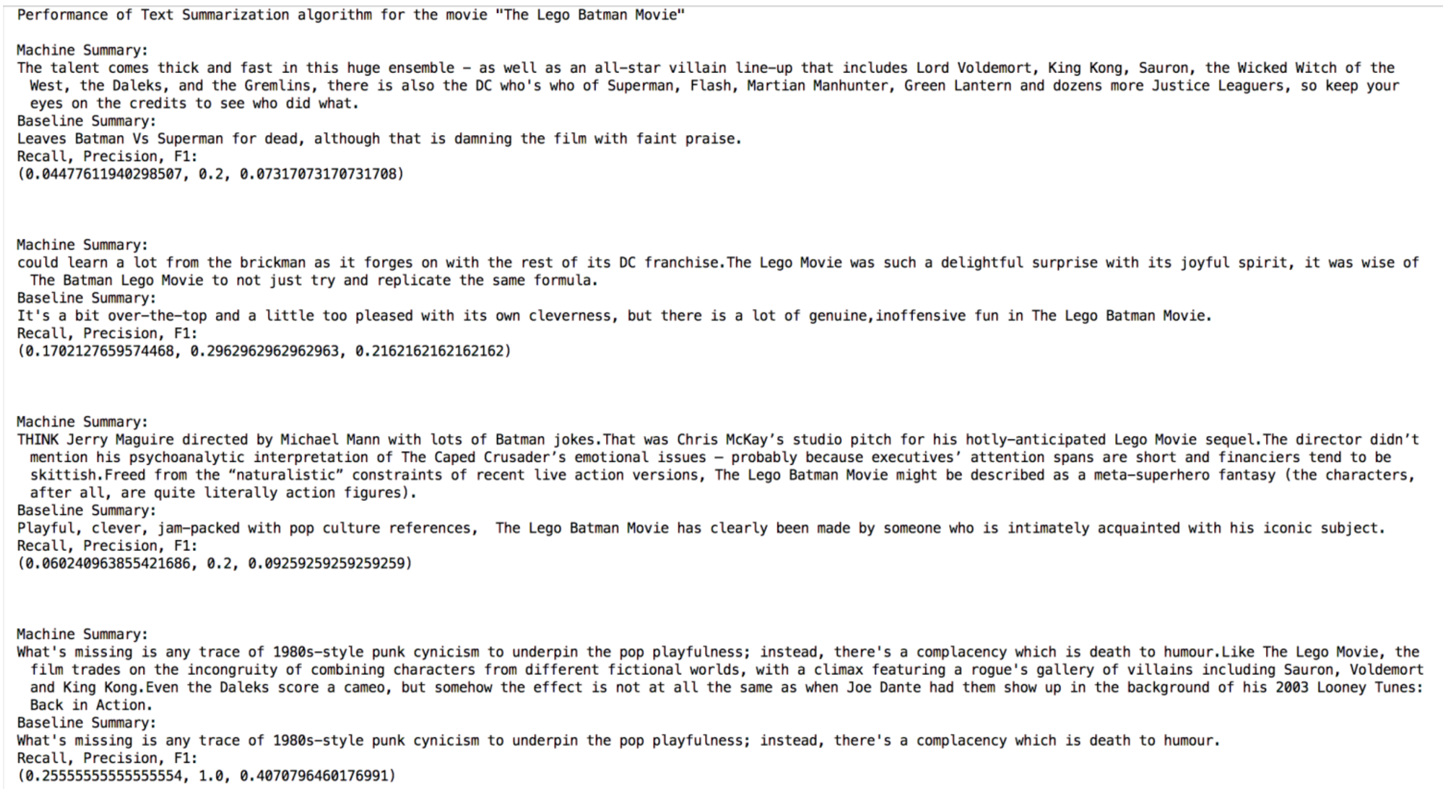
3) ROUGE-S: Measures skip-gram co-occurrence.

And all these rouge-based scores have their own precision, recall and f1 measures.

Baseline:

We will be comparing our system's output with the first sentence of every critic review. Here we assume that the first sentence of our review document is the most important or the one with the most meaningful information required by the user.

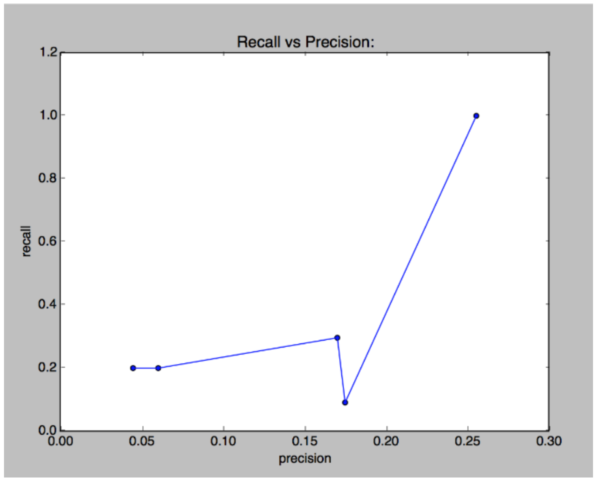
Intermediate/Preliminary Results:



```
Performance of Text Summarization algorithm for the movie "The Lego Batman Movie"

Machine Summary:
The talent comes thick and fast in this huge ensemble – as well as an all-star villain line-up that includes Lord Voldemort, King Kong, Sauron, the Wicked Witch of the
  West, the Daleks, and the Gremlins, there is also the DC who's who of Superman, Flash, Martian Manhunter, Green Lantern and dozens more Justice Leaguers, so keep your
  eyes on the credits to see who did what.
Baseline Summary:
Leaves Batman Vs Superman for dead, although that is damning the film with faint praise.
Recall, Precision, F1:
(0.04477611940298507, 0.2, 0.07317073170731708)



Machine Summary:
could learn a lot from the brickman as it forges on with the rest of its DC franchise.The Lego Movie was such a delightful surprise with its joyful spirit, it was wise of
  The Batman Lego Movie to not just try and replicate the same formula.
Baseline Summary:
It's a bit over-the-top and a little too pleased with its own cleverness, but there is a lot of genuine,inoffensive fun in The Lego Batman Movie.
Recall, Precision, F1:
(0.1702127659574468, 0.2962962962962963, 0.2162162162162162)



Machine Summary:
THINK Jerry Maguire directed by Michael Mann with lots of Batman jokes.That was Chris McKay's studio pitch for his hotly-anticipated Lego Movie sequel.The director didn't
  mention his psychoanalytic interpretation of The Caped Crusader's emotional issues – probably because executives' attention spans are short and financiers tend to be
  skittish.Freed from the "naturalistic" constraints of recent live action versions, The Lego Batman Movie might be described as a meta-superhero fantasy (the characters,
  after all, are quite literally action figures).
Baseline Summary:
Playful, clever, jam-packed with pop culture references,  The Lego Batman Movie has clearly been made by someone who is intimately acquainted with his iconic subject.
Recall, Precision, F1:
(0.060240963855421686, 0.2, 0.09259259259259259)



Machine Summary:
What's missing is any trace of 1980s-style punk cynicism to underpin the pop playfulness; instead, there's a complacency which is death to humour.Like The Lego Movie, the
  film trades on the incongruity of combining characters from different fictional worlds, with a climax featuring a rogue's gallery of villains including Sauron, Voldemort
  and King Kong.Even the Daleks score a cameo, but somehow the effect is not at all the same as when Joe Dante had them show up in the background of his 2003 Looney Tunes:
  Back in Action.
Baseline Summary:
What's missing is any trace of 1980s-style punk cynicism to underpin the pop playfulness; instead, there's a complacency which is death to humour.
Recall, Precision, F1:
(0.25555555555555554, 1.0, 0.4070796460176991)
```

The above image is the screenshot of our system in action, where we have collected data from various critics found on rotten tomatoes about the review of "The Lego Batman Movie" and this is its initial performance, we have used tf-idf overlap scores currently we will be trying out different measures and settings to see which gives us our highest evaluation scores. The recall, precision and F1 scores here are computed using ROUGE-N evaluation metric.

The below graph represents our Precision vs Recall for our initial system.



The stages we are down with are the data collection, implementation of modified PageRank and the Rouge Evaluation System.

Related Work:

Here is the list of Summaries of the Scientific Papers we have referred:

Extractive Summarization Using Supervised and Semi-Supervised Learning:

The author proposed a learning-based approach to combine various sentence features categorized as surface, content, relevance and event. Each sentence feature has its unique contribution. First, deciding which sentence is most important in the text. Second, employing supervised learning classifier to examine features. Finally, re-ranking candidate sentences and extracting the top sentences to summarize the whole text. Also, the author investigated co-training by combining labeled and unlabeled data to perform semi-supervised learning.

Project Similarity/Dissimilarity:

This was the first paper we read to get an idea on how text summarization is done. Only few ideas on the approach are taken by us here no similarity exists between the paper and our project.

Graph-based Ranking Algorithms for Sentence Extraction, Applied to Text Summarization:

Graph-based ranking algorithm is a way of deciding on the importance of a vertex within a graph, by considering global information recursively computed from the entire graph, rather than relying only on local vertex-specific information. In this paper, the author investigated several graph-based ranking algorithms and evaluated their application to unsupervised sentence extraction in a context.

Project Similarity/Dissimilarity:

This gave us the idea of using graphs to represent sentences as nodes in the graph and how we could rank them. Here we take the idea of graph based Ranking algorithms from this paper.

The Evaluation of Sentence Similarity Measures:

In this paper, the author tried fourteen similarity measures which could concluded in three different classes: word overlap, TF-IDF, and linguistic measures, used in the evaluation. Two sentences are considered to be similar if they are a paraphrase of each other, which means they talk about the same event or idea judging from the common principle actors and actions, or if one sentence is a subset of the other.

Project Similarity/Dissimilarity:

Here we learn about different sentence similarity measures so we can utilize them in our system. From this paper we take the different methods of calculating similarities between sentences.

TextRank: Bringing Order into Texts:

The author introduced the TextRank graph-based ranking model for graphs extracted from natural language texts. They investigated and evaluated the application of TextRank to two language processing tasks consisting of unsupervised keyword and sentence extraction, and showed that the results are competitive with other algorithms.

Project Similarity/Dissimilarity:

This is the main paper we take our text summarizer’s algorithm from. The main algorithm discussed in this paper for sentence ranking is used in our system.

LexRank: Graph-based Lexical Centrality as Salience in Text Summarization:

Instead of TextRank, the author considered a new approach, LexRank, for computing sentence importance based on the concept of eigenvector centrality in a graph representation of sentences. This paper assesses the centrality of each sentence in a cluster and extract the most important ones to include in the summary.

Project Similarity/Dissimilarity:
This paper gives us another insight on how to represent edges between our nodes (i.e. the relationship between 2 given sentences). We also implement the same algorithm given here. The paper given above this and this paper are quite similar on how to rank the sentences only the scoring metric for the edges between nodes is different.

Work Distribution:

1) Karthik Shivaram - Data Collection, Similarity Measurements
2) Nikhil Birur - Modified PageRank implementation, Similarity Measurements
3) Zinou Lee - Rouge Evaluation Metrics, Similarity Measurements

Even though the distribution is represented in this manner every stage of the project and the work done has been done with equal contribution from all group members for this project.

TimeLine:

Completed Work:

1) Data Collection
2) Tf-idf Similarity Measure
3) Modified PageRank Algorithm
4) Rouge-N Evaluation Implementation

Remaining Work:

1) Remaining Similarity Measures
2) Data preprocessing (Little bit left)
3) Remaining Evaluation Metrics

The remaining amount of work should be completed by April 18th 2017

If time permits we would try to implement an abstractive summarization approach using sequence to sequence Recurrent Neural Networks.

References:

Links to the above-mentioned Scientific Papers:

http://anthology.aclweb.org/C/C08/C08-1124.pdf

http://www.aclweb.org/anthology/P04-3020

http://www.cis.drexel.edu/faculty/thu/research-papers/dawak-547.pdf

https://web.eecs.umich.edu/~mihalcea/papers/mihalcea.emnlp04.pdf

https://www.jair.org/media/1523/live-1523-2354-jair.pdf

Other Links:

https://stackoverflow.com/