Book Chapter Summarization

Urhum Sheikh, Stuart Harley

CS 4980 - Natural Language Processing

Dr. Sean Jones

May 10, 2022

Table of Contents

Summary

Often in school we have homework assignments where we are expected to read a section of a book and summarize it in our own words. Many people cheat this and go to a site like SparkNotes.com and use the summary they find there. For this project, we wanted to see if we could use NLP to summarize a chapter of a book in a new way so it wouldn't be flagged for plagiarism. Our system takes in a text file that contains a chapter of a book and outputs a summary of that chapter built using the most important sentences in the chapter.

Project Background

Automatic Text Summarization is one of the more challenging and complex problems in the field of Natural Language Processing. Here, we generate a process which consists of and gives a meaningful summary of the text which is obtained from multiple text sources such as books, news articles, blog posts, research papers, emails, and tweets. The world is ever so changing and there is a large amount of textual data now available which makes training such approaches feasible.

There are two broad types of text summarization methods. The first is extractive summarization. These methods rely on extracting important phrases or sentences from a piece of text and stacking them together in a summary. The second type is abstractive summarization. These methods are much more complex, and they generate new sentences from the original text. The sentences generated may not be present in the original text. We go into more detail on the two methods below, but we decided to use an extractive summarization approach due to the extreme complexity of the abstractive summarization method and the limited scope of this project.

1) Extractive Summarization

This method particularly relies on extracting several parts, such as phrases and sentences from a given piece of text and then stack them up together to create sort of a summary. The bigger challenge arises from identifying the right sentences for summarization which is highly important in an extractive method.

There are a few given algorithms out there that help such methods, the one we used in this project is called PageRank. PageRank is primarily used in helping rank web pages in online search results. We can understand the basics of this algorithm by supposing there we have 4 web pages - w1, w2, w3, and w4. These pages would contain links pointing to one another, some pages might not have links at all which are called dangling pages.

- Web page w1 has links directing to w2 and w4
- w2 has links for w3 and w1
- w4 has links only for the web page w1
- w3 has no links and hence it will be called a dangling page

The way these pages will be ranked is through to compute a score called the PageRank score which would be the probability of a user visiting that page.

To capture these probabilities of users navigating from one page to another, there will be a square Matrix M with n rows and n columns and n here is the number of web pages. Each element of this matrix denotes the probability of a user transitioning from one web page to another.

The initialization of the probabilities are explained in the steps below:

1) Probability of going from page i to j, i.e., M[i][j], is initialized with 1/(number of unique links in web page wi)
2) If there is no link between the page i and j, then the probability will be initialized with 0
3) If a user has landed on a dangling page, then it is assumed that he is equally likely to transition to any page. Hence, M[i][j] will be initialized with 1/(number of web pages)

The values of these matrices are updated in an iterative fashion to arrive at the web page rankings. This algorithm can also be used in a similar fashion to rank sentences, which is what we are doing in this project.

2) Abstractive Summarization

Abstractive summarization methods require a much more complex model to function. They often use a Many-to-Many Sequence-to-Sequence (Seq2Seq) model because the problem of text summarization relies on sequential information, aka the long sequence of text to be summarized, and outputs a sequential short summary.

There are 2 main components of a Seq2Seq model, the encoder and decoder. This architecture is used to solve Seq2Seq problems where input and output are different lengths. Long Short Term Memory (LSTM) components are used in the encoder and decoder because they are capable of capturing long term dependencies. The encoder reads in each word one at a time and after the nth input word a feature vector is fed to the decoder. The decoder takes the input and generates the predicted words of the output sequence, given the previous words generated.

The model is trained on a vocabulary of words. In the decoder, the probabilities for each word in the vocabulary to be the next word are generated and the maximum value is chosen. It is difficult for the encoder to memorize long sequences into a fixed length vector so how much attention is paid to every word in the input sequence is modified using an attention mechanism.

Data

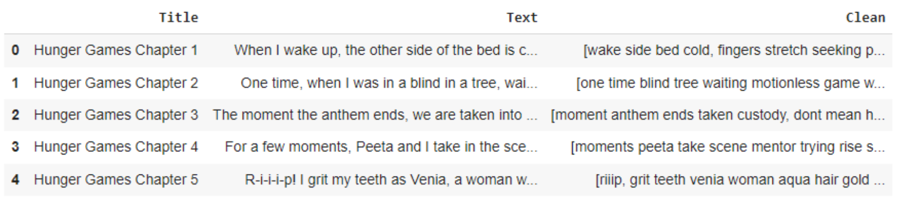
The data that we collected for this project is the collection of chapters from 4 fiction novels: The Hunger Games, Harry Potter and the Sorcerer's Stone, Percy Jackson and the Olympians: The Lightning Thief, and The City of Ember, which we stored in an excel csv file. The authors of these books are Suzanne Collins, J.K. Rowling, Rick Riordan, and Jeanne DuPrau respectively. We did not need to collect summaries of these chapters to train our model on, so once we successfully generated summaries, we used summaries from sparknotes.com and other similar sites to compare our output to.

For our project, we decided to utilize books, in particular fiction novels. This was particularly easy since the books we chose to use are famous enough that finding online pdf versions to copy the text from was relatively simple. The goal of our project was to summarize entire chapters, but because entire chapters have a lot of sentences and words in general, several Harry Potter chapters are not used because the text of the chapter was too long to fit into one excel cell. We decided to simply exclude those chapters for simplicity.

To copy and paste the chapter text into our excel document, we first had to copy the text into a temporary Notepad++ file where we joined all the lines into one long line. If we didn't do this, the text would have been placed into many excel cells separated by every line break. Once the chapter text was in one long line we copy and pasted it into the corresponding excel cell. Separating the novels into chapters is useful because we do not want to train our model on the entirety of the text, so this structuring of the text will make our task much simpler going forward.

The raw format of our data is pure text, since the data being collected are from books and we spend a decent amount of time finding books and then extracting the entirety of the chapters into a cell format. To normalize our data, we removed any punctuation, numbers, and special characters. We also converted the text into lower case so the same words didn't appear multiple times with different cases. We also removed stopwords from the data using the stopwords list from nltk. All of the preprocessing was essential so that non important information was not considered by our model.

Shown below is the cropped output of the pandas head() function showing our data before and after normalization and pruning. The chapters were also turned into a list of sentences.

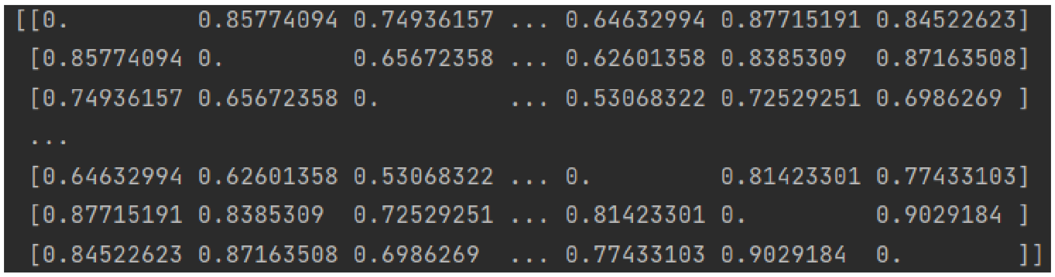


	Title	Text	Clean
0	Hunger Games Chapter 1	When I wake up, the other side of the bed is c...	[wake side bed cold, fingers stretch seeking p...
1	Hunger Games Chapter 2	One time, when I was in a blind in a tree, wai...	[one time blind tree waiting motionless game w...
2	Hunger Games Chapter 3	The moment the anthem ends, we are taken into ...	[moment anthem ends taken custody, dont mean h...
3	Hunger Games Chapter 4	For a few moments, Peeta and I take in the sce...	[moments peeta take scene mentor trying rise s...
4	Hunger Games Chapter 5	R-i-i-i-p! I grit my teeth as Venia, a woman w...	[riiip, grit teeth venia woman aqua hair gold ...

For our original model development we read in our data through a .csv file which contained two columns, the name of the book and the chapter number, and the chapter text. For our final executable we allow the user to specify a txt file containing a chapter to be summarized.

The System

Our system is built using the PageRank algorithm mentioned earlier in this report. Our system first generates a map of the GloVe embeddings of each word in the glove.6B.100d.txt file. This is a list of 400,000 pre-trained word vectors that will be used to calculate similarity scores. Next our system reads in the chapter text and performs all the previously mentioned normalization. After this is complete, each sentence in the chapter has a vector representation of itself created by using GloVe embeddings. Using these vector representations, a matrix of similarity scores is created using these embeddings. The values of this matrix represent how similar each sentence is to every other sentence. A portion of an example similarity matrix is shown below.



```
[[0.         0.85774094 0.74936157 ... 0.64632994 0.87715191 0.84522623]
 [0.85774094 0.         0.65672358 ... 0.62601358 0.8385309  0.87163508]
 [0.74936157 0.65672358 0.         ... 0.53068322 0.72529251 0.6986269 ]
 ...
 [0.64632994 0.62601358 0.53068322 ... 0.         0.81423301 0.77433103]
 [0.87715191 0.8385309  0.72529251 ... 0.81423301 0.         0.9029184 ]
 [0.84522623 0.87163508 0.6986269  ... 0.77433103 0.9029184  0.        ]]
```

Now that we have this similarity matrix, we convert this matrix into a graph. The nodes of this graph represent the sentences and the edges represent the similarity scores between the sentences. Finally, once this graph is built, we apply the PageRank algorithm to this graph to generate the sentence rankings. PageRank forms a cosine similarity matrix which stores the similarity scores of each sentence against one another and ranks them based on these scores. We then take the top 15 ranked sentences to be our chapter summary. We chose the number 15 because we felt this would be enough sentences to hopefully provide a good summary of the chapter but not be too long.

Our prototype system contains a standard python executable file and the GloVe embeddings text file mentioned above. We also include an example chapter text file that can be run, or you can provide your own. However, the chapter text must all be contained on one line of the text file. The system also requires that you have several basic and NLP libraries pre-installed on your Computer. The following packages are required for the python file to be executed:

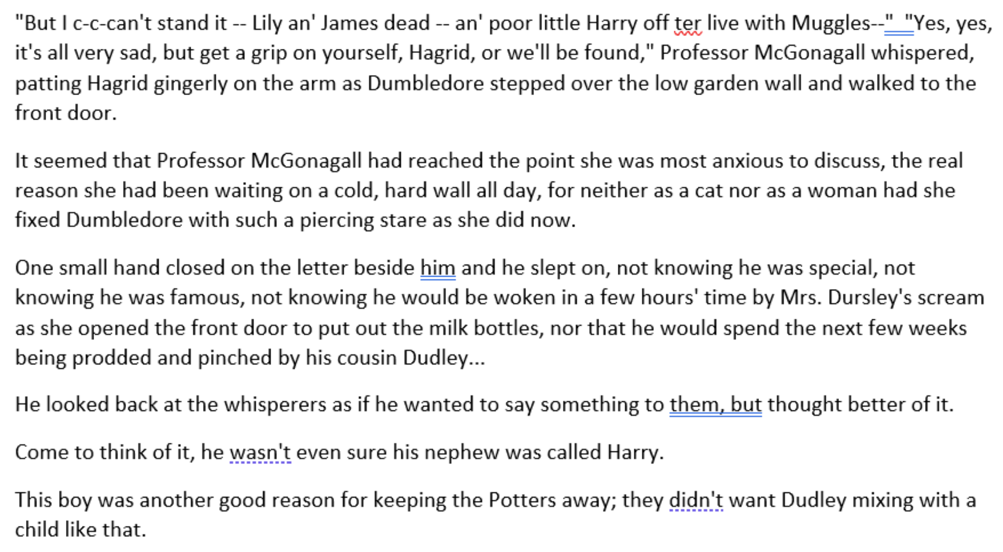
1) Numpy
2) Pandas
3) Time
4) NLTK library
5) Sklearn
6) NetworkX

To run the script, launch the python file, select the appropriate python interpreter, preferably Python 3.10 and then run the script. When prompted, enter the path to the chapter text that you want summarized. The script will then generate a summarization of the chapter based on the provided text file.

Performance

We performed some benchmark timing on the various parts of the system. Creating the word embeddings map from the GloVe file takes 9.9 seconds. The chapter text is normalized in 0.02 seconds. The vector representations of the sentences are created in 0.01 seconds. The similarity matrix of the cleaned sentences is the most time-consuming part of the processing, taking 17.94 seconds. Then the PageRank algorithm takes 0.28 seconds to run. Altogether, the entire process takes ~25-30 seconds to run, depending on the length of the chapter.

The following is the generated summarization from running the script on the first chapter of Harry Potter and the Chamber of Secrets:



"But I c-c-can't stand it -- Lily an' James dead -- an' poor little Harry off ter live with Muggles--" "Yes, yes, it's all very sad, but get a grip on yourself, Hagrid, or we'll be found," Professor McGonagall whispered, patting Hagrid gingerly on the arm as Dumbledore stepped over the low garden wall and walked to the front door.

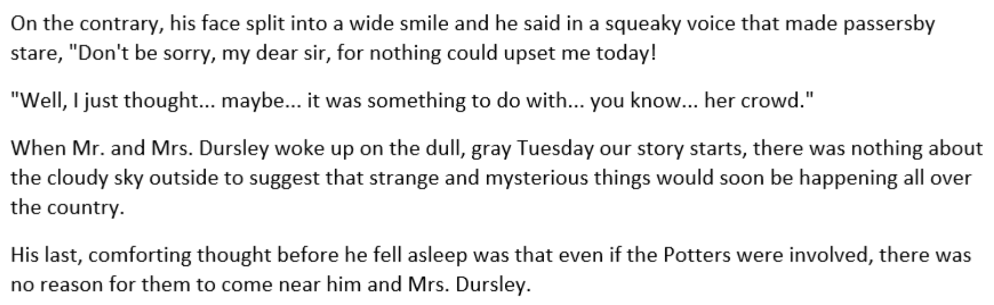
It seemed that Professor McGonagall had reached the point she was most anxious to discuss, the real reason she had been waiting on a cold, hard wall all day, for neither as a cat nor as a woman had she fixed Dumbledore with such a piercing stare as she did now.

One small hand closed on the letter beside him and he slept on, not knowing he was special, not knowing he was famous, not knowing he would be woken in a few hours' time by Mrs. Dursley's scream as she opened the front door to put out the milk bottles, nor that he would spend the next few weeks being prodded and pinched by his cousin Dudley...

He looked back at the whisperers as if he wanted to say something to them, but thought better of it.

Come to think of it, he wasn't even sure his nephew was called Harry.

This boy was another good reason for keeping the Potters away; they didn't want Dudley mixing with a child like that.

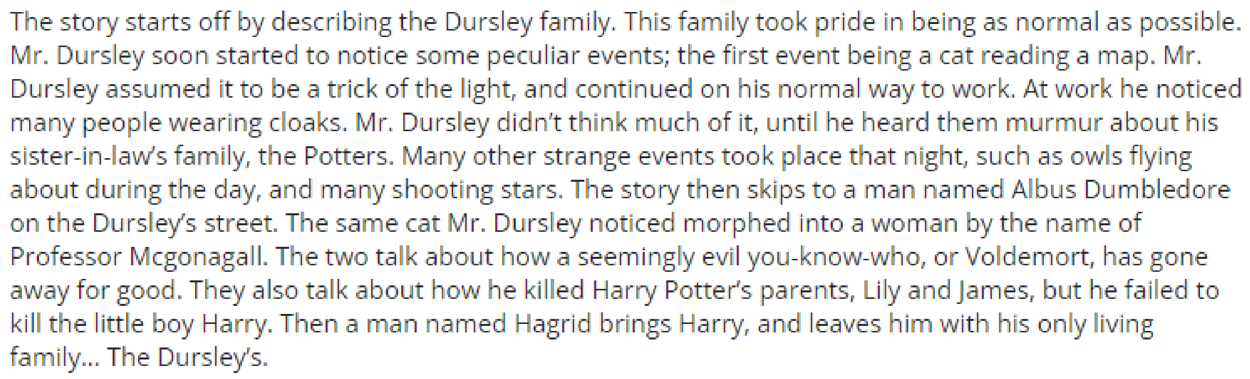

On the contrary, his face split into a wide smile and he said in a squeaky voice that made passersby stare, "Don't be sorry, my dear sir, for nothing could upset me today!

"Well, I just thought... maybe... it was something to do with... you know... her crowd."

When Mr. and Mrs. Dursley woke up on the dull, gray Tuesday our story starts, there was nothing about the cloudy sky outside to suggest that strange and mysterious things would soon be happening all over the country.

His last, comforting thought before he fell asleep was that even if the Potters were involved, there was no reason for them to come near him and Mrs. Dursley.

In comparison, below is a summary of the first Chapter of Harry Potter written by a human taken from the following link: https://mitchharrypotter.weebly.com/blog/chapter-summaries


The story starts off by describing the Dursley family. This family took pride in being as normal as possible. Mr. Dursley soon started to notice some peculiar events; the first event being a cat reading a map. Mr. Dursley assumed it to be a trick of the light, and continued on his normal way to work. At work he noticed many people wearing cloaks. Mr. Dursley didn't think much of it, until he heard them murmur about his sister-in-law's family, the Potters. Many other strange events took place that night, such as owls flying about during the day, and many shooting stars. The story then skips to a man named Albus Dumbledore on the Dursley's street. The same cat Mr. Dursley noticed morphed into a woman by the name of Professor Mcgonagall. The two talk about how a seemingly evil you-know-who, or Voldemort, has gone away for good. They also talk about how he killed Harry Potter's parents, Lily and James, but he failed to kill the little boy Harry. Then a man named Hagrid brings Harry, and leaves him with his only living family... The Dursley's.

As far as the performance is concerned for our model, we are comparing the summaries by hand with visual analysis. Clearly our model doesn't do a good job at making a summarization of the chapter that makes sense. Due to the PageRank algorithm which ranks the important sentences, only existing sentences from the chapter can be used, which do not fit together nicely to form an understandable summary.

Future Work

On paper, our project idea seemed cool. We had the right tools available to succeed and we did appropriate research to investigate algorithms that would help us achieve our goal of constructing a model that would generate constructive text summarization for books where feeding in a chapter would result in a meaningful chapter summarization of which could be used for school assignments.

Unfortunately, since our current model only selects popular extracted sentences and ranks them to generate text, our summaries do not end up being meaningful. It doesn't generate coherent sentences that make sense when grouped together with other sentences. It is very random and just not what we were expecting. We hypothesize that this type of system is not applicable for entire chapter summarization because there is just too much text, much of which is dialogue or unimportant information, and some of this is bound to end up in our summary. We believe this sort of model would’ve been more useful for reading relatively small articles where sentence sequence is not as important, and sentences in the article would be more informational, rather than trying to tell a story.

For future work, our project would require a different approach. Instead of using extractive text summarization techniques like the PageRank algorithm, we could use an abstractive summarization Many-to-Many Sequence-to-Sequence (Seq2Seq) model. Ideally, because this type of model generates novel sentences instead of being limited to existing sentences, it would have been able to better summarize book chapters.

We wouldn’t say this entire project was a waste because we are able to conclude that extractive summarization methods are not good for summarizing works of fiction. If we changed our methodology here to using abstractive summarization techniques, we believe we could generate summaries closer to what we envisioned in our original vision statement.

Lessons Learned

Working on this project gave us a general overview of how summarization techniques are applied and what kind of algorithms are associated with them. It is generally much more difficult than anticipated because there are various techniques out there and our expectation was to have at-least a semi working model that would generate meaningful summarization for large text which in our case was books.

Our initial data gathering and the process associated after all this work seemed like we would generate a model that would achieve our expected goals. Unfortunately, the PageRank algorithm did not work well for this project so we ended up with lackluster results. Our current model is more suited for relatively small articles with sentences that highlight important information instead of sequential sentences that when placed out of order, do not make much sense. Overall, we learned about how summarization techniques are applied and work, even if our final model was disappointing.

References

https://nlp.stanford.edu/projects/glove/

https://www.analyticsvidhya.com/blog/2018/11/introduction-text-summarization-textrank-python/

https://www.analyticsvidhya.com/blog/2019/06/comprehensive-guide-text-summarization-using-deep-learning-python/