CSCE 5290: Natural Language Processing
Project Increment 1

Link to github repo:
https://github.com/RachaelC358/NLP_Group-14_Project

Project Description:

1. Project Title and Team Members

Title

Text Summarization of Sports Recaps for Game Analysis

Team Members

- Nathan Adams
- Jacob Benz
- Rachael Carpenter
- Zachary Smith

2. Goals and Objectives:

Motivation

Most major sports have a number of outlets which cover each team throughout their respective seasons. These sources of commentary and statistics are useful documents for understanding how well a team is performing in a season and may be the only resource that a fan might see for a particular game or event. For instance, fans of baseball will have 162 games a year to keep up with for a single team. While there are certainly diehard fans who will watch every minute of every game, most people will often rely on post-game recaps to understand what happened in a game they did not watch. Even in sports with fewer games in a season, like college football, there is no shortage of opinion articles or game breakdowns.

Noting that news for sporting events far exceeds the live coverage, a method for viewing recaps or articles in a condensed, bite-sized manner would be useful for someone trying to catch up on the latest game. By summarizing recaps, there is potential for additional analysis to be done in a quick manner as well. For instance, sentiment analysis might be a function that is added on after

the text summarization is complete. This is dependent on how much time is needed to complete the initial functionality, however.

Significance

Automatic text summarization can help save time in people's everyday lives by allowing them to make informed decisions with less effort. Manually creating summaries of large sets of data is so inordinately time consuming that it would be almost impossible to do without machine learning.

Sports, in general, have a massive following and as result there are almost too many resources to keep up with. Condensing the corpus of information for a particular team could help sports maintain interest as a season goes along. Many fans stay engaged in the early days of a new season but then later disengage as the games pile up. A programmatic way to deliver fast news could reduce fan falloff and give exposure to a larger variety of sources. This technique has been used for political or world news already to varying effect. Our idea is to essentially provide the analog of an academic article abstract in the world of sports. Another similar idea is that of a book summary on the back cover. Once a user reads the summary, it may prompt them to consume the article in its entirety.

As mentioned before, our idea may be extended beyond summarization. We may use our summaries and a simple sentiment analysis to provide quick information and context in a single package. This yields a result that not only explains what happened but might indicate the tone of the report. By collecting a number of sources for the same event, a fan could get some of the same information as they would have watching the game live.

Objectives

Text summarization has been done in multiple different ways, but we want to implement an accessible solution that caters towards sports coverage. Web scraping documents combined with utilizing neural networks and other NLP techniques will allow us to provide a brief summarization of any given article.

In our final iteration, our project could be hosted on a webpage, where the user can input a URL and our application will return a summarized version of the article. Since we plan to use this specifically for sports, other information such as scores, teams, and important players/plays can be extracted, formatted, and displayed for the user as well.

While the primary objective is summarization, we want to attempt to gather as much information as possible so that our project can easily be extended upon each module's completion. If we can

develop good web scraping tools, pull the score for each game, and then store that information, we can use that data to not only summarize text recaps, but to also create a database of statistics. This is a secondary objective but one which naturally follows our primary summarization goal.

Features

A primary feature of this project is its ability to gather a large amount of text information from varying sources on the internet. This is currently planned to be achieved via the earlier stated technique of web scraping, which, in addition to its above outlined importance, will play a major role in the data collection portion of the research and development.

One obvious feature of our design plan is the ML aspect, which includes a model/technique that will succinctly coordinate text processing and creation. As it currently stands, there are many paths forward to go about such a design, and details such as which exact model, method of normalization, and smoothing algorithm can be determined as we progress forward. Currently, the probable method we plan to utilize will involve a variation of a neural network.

Ideally the user interface component of the project should be as simple as possible, both in terms of creation and UI/UX design. This will involve an easy-to-read website with minimal extraneous features.

Increment 1 Guidelines

For all the features developed for this increment, write a documentation describing the design, implementation, testing, and deployment (including descriptions and screenshots).

Related Work (Background)

In a 2018 study, researchers proposed a methodology using Restricted Boltzmann Machine (RBM) extractive summarization methods on sports news articles. Researchers hope that finding a method for automatic text summarization will save time and prevent news consumers from being overloaded with information. This involved a generative stochastic artificial neural network. Extractive summarization is used to be able to keep sentences that express the main ideas of the text so other sentences can be removed without losing important information. The steps used to process the text started with preprocessing, extraction, enhancement, and summarization (Priyadharshan).

Dataset

The dataset involved in this project is atypical; as the primary purpose of our analysis is summarization, and our theme focuses on sports articles as a background, any article that fulfills those requirements satisfies the purposes of our approach. Additionally, as a major objective of the assignment is merely comparison of the models, the exact length of what is summarized after a certain length, maybe 1000 words, becomes trivial. One such article covers a recent Cowboys/Eagles game, via 'bleedinggreennation.com'. This article is not especially lengthy in size, but contains enough specific information about both the game's outcomes and specific plays to demonstrate certain metrics in the models later on.
Summarization requires somewhat of a complex approach at first, but many of these steps are made simple using the library *newspaper*, which is able to extract the article from a given URL, given in-library parsing() and nlp() functions. Additionally, in one such model *Sumy* provided similar resources, providing an HTML parser given a URL.

Detail Design of Features

We want to evaluate various text summarization models for comparative purposes. The models that were chosen for implementation are prebuilt and are as follows:

1. Gensim
    - Extractive summarization
    - Uses TextRank to determine sentence importance.
    - Sentences with similar words recommend each other and form connections in a connected graph. The sentences in the graph with the most connections are ranked as more important.
2. SpaCy
    - Extractive summarization
    - Uses SpaCy's corpus and stopwords
    - Uses Laplace smoothing
3. Scikit-Learn
    - Extractive summarization
    - Uses lemmatization and a TF-IDF vectorizer
4. Sumy
    - Extractive summarization
    - Employs a stemming function to reduce the number of terms to roots
    - Uses sentence ranking
    - Summarizes to a user-specified number of sentences

The goal is to take the outputs from each model and determine a way to compare their results quantitatively. There are a number of summarization metrics that have been used for other models, including Rouge and Bleu. However, we may need to theorize another method for model

comparison. This will be the focus of our second portion of the project. Additionally, we may consider running all of the models on the same data, consecutively to see if combining methods results in a more robust result. Experimentation will be necessary to determine the efficacy of this technique.

Analysis

Sumy

Sumy's differentiation from the other methods used in our project comes in the form of its stemmer functionality. Unlike the other techniques, words are stemmed to their roots and then tallied for term frequency. With this, the sentences can be ranked by a cumulative sentence importance measurement that is based on how common all of the words are within the sentence as a whole. With the sentences ranked in this manner, the top results are returned based on the number of sentences requested by the user.

SpaCy

SpaCy's results are certainly understandable and accurate, however are very similar to the article itself. The approach is mostly algorithmic in nature, and unlike its more complex or ML counterparts, lacks understanding of sentence structure and context, hence its similarity to the source material.

Scikit-Learn Implementation

The Scikit-Learn model generated a fairly accurate summary for it's method of selecting sentences. The frequency of lemmatized tokens managed to extract the majority of the important moments from the article. Because it is extractive, only words from the article are going to be used in the summary, but this combined with the easily variable summary length yields a fairly versatile summarization algorithm.

Gensim TextRank Algorithm

TextRank works by selecting sentences that have the most words in topics in common with other sentences throughout the article. Looking at the output summary, we can see the sentences chosen for the summary tend to contain topics that occur multiple times throughout the article, as well as a high number of topics per sentence. This is a successful method for summarization because sentences highly ranked by this algorithm will likely be sentences that connect many subjects or ideas found within the article.

Implementation

Four separate models were used to test summarization. Extractive models take words or sentence fragments from the original text based on a number of factors, and add them to the final

summary. Abstractive models construct some form of language structure to then summarize the original meaning of the article. In our current implementation, all of the models are extractive in nature. However, we would like to implement an abstractive method for contrast.

Scikit-Learn Implementation

Scikit-Learn's libraries and a TF-IDF vectorizer are used to create a summary. TF-IDF stands for term frequency-inverse document frequency. This means that it weighs the amount of times the term appears in a document against the importance of that word in the document. This softens the importance of stopwords such as "is" and "the". The Sklearn implementation uses newspaper3k libraries to get the text from any URL. Then it parses the original text into tokens. All of the punctuation marks get removed, and the tokens are then lemmatized. Stopwords are left in, since TF-IDF has a built-in method for dealing with these words. In order for a token or tokens to get added to the final summary, the TF-IDF average result for that token has to exceed the set threshold value. The threshold is a combination of the total appearances in a document versus the number of documents, multiplied by a self-set value. The custom value allows this implementation to be flexible on the length of the summary returned. All of this was based off of the referenced Medium article by Lucía LLavero.

Gensim TextRank Implementation

TextRank uses graphs to compare sentences by connecting sentences with similar concepts or meanings to show commonality. So, sentences that have similar meaning will recommend each other to be ranked as important. The sentences with more connections or with heavily weighted connections are assumed to be the most important sentences in the text (Mihalcea).

Sumy Implementation

Sumy is an extractive text summarization tool developed by Mišo Belica as his graduate work thesis. The tool itself is very simple to set up and retrieves a single document for summarization. The document is subsequently broken up into paragraphs, sentences, and words via NLTK's tokenizer. Included in the tool are preprocessing features, a word normalizing stemmer, and the summarizer itself. The great thing about Sumy's implementation is that each of these features are separated and can be used as desired. Finally, the user can determine the length of the summary by telling the functions the number of sentences that should be returned after summarization.

SpaCy implementation

This approach, largely inspired by NumpyNinja's article, depends largely on the stopwords and punctuation defined by SpaCy, which may vary from the other methods. A large part of the *summary* function is actually just data cleaning as it pertains to these SpaCy definitions. The article is fed through a laplace smoothing algorithm, then normalized according to the max

frequency and ranked as such. By definition then, this approach is *extractive*, and largely focuses on which important keywords appear the most and to what context they appear in. The *per* argument controls the length of the generated summary. However, a large section of the remaining work of *summary()* is hidden behind heapq's *nlargest* function, which sorts the scored sentences by largest, and selects *per* amount of sentences for appearance in the final generation.

Preliminary Results

The same 7,200 word article was used to compare the outputs of our respective models. The results varied in length and accuracy, but their viability depends greatly on the user's desired outcome. Consequently, a general and broad method for ranking summaries needs to be developed.

Scikit-Learn Implementation:

```
The Eagles used an explosive 20-point second quarter to win. Gardner-Johnson and one by cornerback Darius Slay. Cornerback James Bradberry deflecting a pass intended for Gallup. Safety C.J.
Dickerson twisted, then buried Golston. It was the most significant play in shaping the tenor of the game. Johnson, who had done a good job against Parsons, did not block him at all. Hurts had
him open and overthrew. The Eagles recovered the fumble at the Dallas 42 with 2:46 to play. The flag gave Dallas a first down and set up the Ferguson score. Then Cox watched Elliott on the back
end run right by. Detwiler in February 2006, which appeared on SportsCenter.
```

Sumy

```
The Eagles remained the only undefeated team in the NFL by finding a way to win again, getting by the Dallas Cowboys, 26-17, after almost blowing a 20-point lead Sunday night at Lincoln Financial Field.
The Eagles' defense threw some heat at Cowboys' backup quarterback Cooper Rush, who completed 18 of 38 for 181 yards and a touchdown, while throwing three interceptions, two by safety C.J.
On third-and-nine at the Dallas 26, cornerback Darius Slay stepped in front of a Rush pass intended for Michael Gallup with 5:14 left in the second quarter, which eventually led to Jake Elliott 's second field goal.
Hurts finding Brown on third-and-three at the Dallas 15 with his second TD reception of the season and a 14-0 Eagles' lead with 4:07 left in the half.
It was a great designed play, which caused Parsons to momentarily hesitate at misdirection, and allowed Hurts to pitch a short pass to Brown for an 11-yard gain.
Defensive end Josh Sweat closing down on Ezekiel Elliott on the Cowboys' first drive, on a third-and-two at the Dallas 24.
Linebacker Haason Reddick not being able to get off the block of Cowboys' right tackle Terence Steele on the first play of the game, when Lamb took an end around for eight yards.
Defensive tackle Fletcher Cox being totally taken out by Steele, who had a great game, on the Elliott third-quarter touchdown run.
Linebackers Kyron Johnson and Nakobe Dean losing outside contain on KaVontea Turpin's 63-yard kickoff return that led to the Cowboys' only points of the first half.
In 2006, he was nominated for an Emmy Award for a special project piece for ESPN.com called "Love at First Beep." He is most noted for his award-winning ESPN.com feature on high school wrestler A.J.
```

Gensim TextRank

```
The Eagles' defense threw some heat at Cowboys' backup quarterback Cooper Rush, who completed 18 of 38 for 181 yards and a touchdown, while throwing
three interceptions, two by safety C.J. Gardner-Johnson and one by cornerback Darius Slay.
On third-and-nine at the Dallas 26, cornerback Darius Slay stepped in front of a Rush pass intended for Michael Gallup with 5:14 left in the second
quarter, which eventually led to Jake Elliott's second field goal.
Hurts finding Brown on third-and-three at the Dallas 15 with his second TD reception of the season and a 14-0 Eagles' lead with 4:07 left in the half.
On the Eagles' second drive, on fourth-and-three at the Dallas 38 with 3:38 left in the first quarter.
Gardner-Johnson and Epps both whiffing on Dallas tight end Jake Ferguson on the Cowboys' seven-yard, fourth-quarter touchdown reception with 14:39 to
play, putting Dallas within 20-17.
The play led to Ezekiel Elliott's 14-yard touchdown run with 8:19 left in the third quarter.
```

SpaCy

```
C:\Users\zacha\anaconda3\python.exe C:\GitHub\NLP_Group-14_Project\models.py
On second-and-14 at the Eagles' 24 with :15 left in the third quarter, Reddick failed to take down Elliott, who reached the Eagles' 12 on the last play of the third quarter.

Hurts' 21-yard completion to back-up tight end Jack Stoll on the Eagles' second drive, which brought the ball to the Eagles' 43 on second-and-eight.Gardner-Johnson and Epps both whiffing on Dallas tight end Jake Ferguson on th

On the Eagles' first score, left guard Landon Dickerson manhandling Dallas' tackle Chauncey Golston, creating a running lane for Sanders to score the Eagles' first touchdown.The Eagles' defense threw some heat at Cowboys' back

Hurts finding Brown on third-and-three at the Dallas 15 with his second TD reception of the season and a 14-0 Eagles' lead with 4:07 left in the half.

A Dallas' neutral zone infraction set up Sanders' five-yard TD run on the first play of the second quarter.Defensive tackle Jordan Davis fighting off Dallas center Tyler Biadasz to stop Tony Pollard for a two-yard gain on firs

Brown dropping the Jalen Hurts' first pass of the game at the Eagles' 35.

The defended pass led to an Elliott 51-yard field goal and a commanding 17-0 Eagles' lead.

Jalen Hurts finished by completing a modest 15 of 25 passes for 155 yards and two touchdowns, while A.J. Brown caught five passes for a team-high 67 yards and his second touchdown of the season, and Miles Sanders rushed for 71

Slay batted down another pass on second-and-10 at the Eagles' 41 with 1:17 to play.

The following play Hurts hit DeVonta Smith with a touchdown pass, padding a slim lead into a 20-17 cushion with 7:02 to play.

The play led to Ezekiel Elliott's 14-yard touchdown run with 8:19 left in the third quarter.

Linebackers Kyron Johnson and Nakobe Dean losing outside contain on KaVontea Turpin's 63-yard kickoff return that led to the Cowboys' only points of the first half.

So far this season, they've scored 112 points in the second quarter, and just 14 points in the first, 17 in the third and 18 in the fourth.

It extended the Eagles' drive and protected the slim 20-17 lead.

The Good

On second-and-nine at the Dallas 29, Hurts finding Brown for a 22-yard gain and a first down at the Dallas seven with 7:32 to play.
```

Project Management

Implementation status report

▪ Work completed:

- Four extractive text summarization models

• Description

- These four models successfully created summaries of our test article. Each one had varying answers as well as customizable summary length for a more versatile implementation.

• Responsibility (Task, Person)

1. Nathan Adams
    - Implemented summarization using Sumy, an extractive technique
2. Jacob Benz
    - Implemented summarization using Scikit-Learn, an extractive technique
3. Rachael Carpenter
    - Implemented summarization using Gensim TextRank algorithm, an extractive technique
4. Zachary Smith
    - Implemented summarization using SpaCy, an extractive technique

• Contributions (members/percentage)

1. Nathan Adams
    - 25%
2. Jacob Benz
    - 25%
3. Rachael Carpenter
    - 25%
4. Zachary Smith
    - 25%

▪ Work to be completed

- Build an abstractive, pipelined model
- Evaluation method to decide the best model
- Website for user interaction

• Description

- **This final model will combine our knowledge from the previous four, and generate a meaningful summary based on sentence structure and importance. We also plan to pipeline this model with another one and analyze the effectiveness.**
- Existing methods such as ‘Rouge’ and ‘Bleu’ score can be used to estimate effectiveness of text summarization, but these both require human responses in order for a score to be calculated. We need to devise our own method of scoring the accuracy of our text summarization models.
- We are going to implement a user interface for our model/models to be used. Any URL can be given and, based on how well we score each of our models, a specific one will be chosen to return the user a summary of the given URL.

• Responsibility (Task, Person)

1. Nathan Adams
2. Jacob Benz
3. Rachael Carpenter
4. Zachary Smith

• Issues/Concerns

- Since established metrics are limited for comparing our models, we will have to approach this problem creatively.

References/Bibliography

“Gensim: Topic Modelling for Humans.” *Radim Å˜EhÅ¯Å™Ek: Machine Learning Consulting*,

https://radimrehurek.com/gensim_3.8.3/auto_examples/tutorials/run_summarization.html.

Huang, K. H., Li, C., & Chang, K. W. (n.d.). Generating Sports News from Live Commentary: A Chinese Dataset for Sports Game Summarization. *Tencent AI Lab*. https://aclanthology.org/2020.aacl-main.61.pdf

Mihalcea, Rada, 1974- & Tarau, Paul. TextRank: Bringing Order into Texts, paper, July 2004; [Stroudsburg, Pennsylvania]. (https://digital.library.unt.edu/ark:/67531/metadc30962/: accessed November 5, 2022), University of North Texas Libraries, UNT Digital Library, https://digital.library.unt.edu; crediting UNT College of Engineering.

Priyadharshan, T., & Sumathipala, S. (2018, December). Text Summarization for Tamil Online Sports News Using NLP. *2018 3rd International Conference on Information Technology Research (ICITR)*. https://doi.org/10.1109/icitr.2018.8736154

Sblendorio, D. (2022, January 8). *How to do text summarization with deep learning and Python*. ActiveState. Retrieved October 7, 2022, from https://www.activestate.com/blog/how-to-do-text-summarization-with-python/

USMNT scoreless draw against Saudi Arabia adds more concerns before World Cup. (2022, September 28). Stars and Stripes FC. Retrieved October 7, 2022, from https://www.starsandstripesfc.com/usmnt-friendly/2022/9/28/23377078/usa-usmnt-scoreless-draw-friendly-saudi-arabiarecap-world-cup

Company, L. L. (2020, May 19). *Building a text summarizer in python using NLTK and scikit-learn class TfidfVectorizer*. Medium. Retrieved November 5, 2022, from https://medium.com/saturdays-ai/building-a-text-summarizer-in-python-using-nltk-and-scikit-learn-class-tfidfvectorizer-2207c4235548

Note: The references must be linked in the document.

Submission Guidelines:
1. Submit your source code and documentation to GitHub and Canvas.
2. The GitHub link should be provided in the document.
3. Source code has to be properly commented.
4. The documentation should include the visualization of preliminary results.
5. Submit a brief demo video showing your source code and project tasks with a voice over explaining your work through the submission link.
6. The similarity score for the document should be less than 15%.