Generic Summary Generation to Produce a Highlighted Version of Documents

Juan E. Junco
Earlham College
Richmond, Indiana
Jejunco20@earlham.edu

ABSTRACT

Identifying the most relevant text information from online texts has become a standard task now that data is easily accessible. Consequently, summary generation has gained significant importance by reducing the amount of text a person has to read to get meaningful information from the text. The underlying field for this process Natural Language Processing (NLP). NLP enables algorithms to understand and rewrite a text based on the original information. In this proposal, I will explore how a Convolutional Neural Network (CNN) can create a new approach for visualizing the critical aspects of a text. Instead of generating a shorter text, the output will be a version with the essential elements highlighted.

KEYWORDS

Summary Generation, Natural Language Processing (NLP), Machine Learning (ML), Convolutional Neural Networks (CNN), Artificial Intelligence (AI).

1 INTRODUCTION

The definition of summary generation I will use in this paper resembles the definition stated by H. Lie. he refers to summary as a condensation of the main ideas in an article and defines it as a text reduced to its main points [8]. The massive number of digital texts on a single topic increases the difficulty of identifying relevant information. "Summarization is important in some context to help people understand facts or to gain knowledge." [16]. Summary generation utilizes ML, statistical analysis, and NLP strategies [11]. In this paper, I will explore how using tools for summary generation along with a CNN allow the creation of new output. The work was inspired by Yadav et al. (2018) [7] and is guided by the research question, Is a CNN effective to generate a highlighted document version for users that identifies the most relevant parts of the text that convey the document's meaning?

The idea that key portions of a document can be highlighted instead of summarizing has been present for quite some time. Among the motivations to do so, researchers argue that, highlights appear within their context (unlike a summary), and the impact of 'bad" highlights is of much lower consequence than 'bad" summaries [18]. The importance of the proposed software is that it highlights the most relevant sentences in a text. The software would allow the user to identify the fundamental ideas of a text using the author's words. Moreover, this approach avoids the process of sentence generation or reconstruction to generate the summary that is used in some of the work surveyed for this review. This approach uses the same methods of summary generation though a CNN of Yadav et al. and combines it with Spala et al. methods to highlight specific parts of the text. Currently, most methods for summary generation extract relevant sentences or words and categorize them based on statistical analysis, and finally reach the point of summary reconstruction. In addition, every relevant sentence is glued together using punctuation marks and connectors to build a coherent summary.

When presented with a summary, the user is faced with a choice: either rely on the summary or take the long approach and skim through the full text. As a solution, this study aims to address that limitation by successfully identifying the general ideas of a text and using them to generate a highlighted version of the text that spotlights the key points. The user can now identify the same information as in the generated summary but relying on the veracity of the words written by the author. The models will be evaluated using the accuracy Recall-Oriented Understudy for Gisting Evaluation (ROUGE) scores as a measurement for relevance of text as suggested by the work by Baldwin et. al [1].

The remainder of this paper is organized into six sections. Section 2 provides a brief overview of the related work. Section 3 describes the design of the project. Section 4 outlines the evaluation methods. Section 5 shows the results and Section 6 contains the conclusion.

2 BACKGROUND AND RELATED WORKS

This section introduces existing research on two widely-used methods for text summarization and similar work that also highlights significant aspects of a text. The summarization methods explored are the statistical approach and the convolutional neural networks (CNN). The work surveyed for this proposal generates a new summarized piece of text using one of those two methods, but very few generate a highlighted version. The generation of that document will be the objective in my senior capstone project.

There are different types of summaries. According to Berger and Mittal [2], summarization is a field divided into two different ways of generating the desired extraction of information: Generic summaries and query-relevant summaries. This project will focus on the former type of summary. Generic summaries allow understanding of information without considering what specific pieces the reader might be hoping to extract from the article. In other words, it is much like 'the abstract in a paper, designed to distill salient points" [2]. The outcome of such analysis is a new text that semantically joins relevant aspects into a new text. This kind of summary allows users to understand the main points of the text independent of any query.

In this paper, I will use generic summaries, but there is a direct contrast that generates a summary based on the user's query. "User Focused Abstracts, i.e., abstracts relating information in the document to a particular user interest" [10]. The summary reflects the importance of individual pieces of the text that match the text query made by the user. Basically, it spotlights the most important aspects of the text based on what the user is looking for.

This section will briefly explore related work that attempted to highlight the most important aspects of a text. Then, I explore the utilization of a NN and how efficient it is to generate summaries based on sentences from the text in a categorized way. Finally, statistical analysis has a broad range of operations to achieve a hierarchy of either words or sentences in the text. When the latter approach is implemented correctly, it has a similar and, in some cases, higher accuracy than using a NN.

2.1 Highlighting Text

Spala et al. implemented a survey study that presented two different sets of human candidates with the same text version. One group was shown an unannotated text, and annotators were asked 'to highlight sentences that would make document comprehension easier and faster for another naive reader"[17]. The second group was presented with two already highlighted document versions. Participants had to vote on every highlight displayed on the document. Once they reached the document's end, they had to rate the two versions of the highlights. Spala et al. used the interaction of human users to come up with a correct highlighted version of documents. Spala et al. is an excellent example of how a highlighted version of a document is possible but has yet to be fully automatized.

In a second approach, Turney tested different benefits of extracting keyphrases from documents, among them the benefit of having a highlighted version of the text [19] . P. Turney treated a document as a set of phrases, which a learning algorithm learns to classify as positive or negative examples of key phrases. Turney's first set of experiments applied the C4.5 decision tree induction algorithm and the second set of experiments applied a GenEx algorithm specifically for this task. Highlighting was included by a direct comparison between the GenEx algorithm and Verity's Search 97 text retrieval system [5]. Verity's Search 97 produces a summary with highlighted key phrases embedded in the sentences. Turney utilized a search-based optimization technique based on Genetics and Natural Selection principles. This lies outside of all the other summarization techniques surveyed in this proposal; the aim of P. Turney was not specifically to produce readable documents for the user but to test the importance of key phrases.

2.2 Statistical Approaches

Statistical approaches summarize a document using statistical features of the sentence, such as title, the location and, term frequency, assigning weights to keywords (keywords are words of the title in the text) and then calculating the score of the sentence. Finally, the software selects the highest-scoring words and adds them to the summary [3].

Baldwin and Morton [1] implemented an information retrieval algorithm based on the probabilistic analysis. The main objective was to find all the possible sentences in the text until the query was 'covered," meaning that a sentence contains information related to the query. The method used two features, first finding the probability for a word in the document to match the word on the query. Therefore, words were assigned a number based on how similar they were to words from the query. The words with the highest calculations were further analyzed with NLP techniques, including:

- Entity Recognition
- Tokenization
- Sentence Detection
- Part of Speech Tagging
- Morphological Analysis
- Parsing
- Argument Detection

Baldwin and Morton categorized the sentences using co-reference chains, pieces of text with any common sub-sequence of words from the query. After categorizing sentences based on resemblance with co-reference chains, those with the highest probability were organized in ascending order and added to the final summary. The following query showcases the possible sentence selected from a text using their algorithm:

Query: What evidence is there of paramilitary activity in the U.S.?

Summary: Last month, the extremists used rocket-propelled grenades for the first time in three attacks on police and paramilitary units

Munot and Govilkar performed a comparative study using other statistical approaches [11]. In the study they show how 'abstractive text summarization method generates a sentence from a 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." Their software identifies items from the original text and then produces a new document using fewer words.

Software that uses NLP techniques has a very effective rate of summary (the appropriateness of the sentences it produces as output). In both articles mentioned above, the testing showed more than 90 percent accuracy with human generated summaries. Each sentence in the text is compared with the reference summary and measures the overlapping percentage of words between them. In terms of the implementation of such software, both articles classified sentences and used mathematical analysis to break down the measured categories.

2.3 Neural Networks

'An artificial neural network consists of an input layer of neurons (or nodes, units), hidden layers of neurons, and a final layer of output neurons" [20]. In summary generation, the input is a pre-processed version of the text that can be given to a neural network. An NN is a powerful pattern classification tool. Pattern classification is important to summary generation, as the association of related words allows the reduction in length of a text [13].

Kaiaikhah et al. [7] by creating different features and then manipulating the hidden layers to output a summarized version of the text. The point of uniqueness of this research lies in turning sentences into vectors. Each sentence was evaluated using seven different criteria. Each criterion became the input of the NN, meaning that the information had already normalized to some extent. Kaiaikhah et al. first used already categorized summaries to train the NN. The hidden layer calculations allowed the NN to decide which feature had the most impact on the summarization process by pruning. Every feature that contains the highest resemblance to the query is gathered together for the last stage of the process, the document reconstruction.

A second relevant example of using a NN to produce a summary was proposed by Sinha et al. [15] implemented a successful Neural Network (NN) model that generated very accurate summaries of online texts with an output accuracy of 95 percent compared to human-written summaries. The main point of their approach is to optimize news articles. News articles encapsulate the topic in the title, but usually the story's context requires more than just the title to be understood. The NN requires a numerical representation of the input to perform calculations. Sinha et al. used a vector representation of words. They fed the sentences as input to the word2vec [4] model that provides vector representation for words of the English language. Once the calculations are done, there is a final activation function in the output layer. This allows the NN to gather sentences with the highest measure of belief to be included in the summary. In other words, the highest numbers are added to the summary. Finally, after extracting the relevant points, Sinha et al. used ROUGE scores as the evaluation method to test for accuracy. This technique compares a human-generated text summary with the output summary of the NN using n-grams. The summaries had an average accuracy score of 95 percent compared to human-performed summaries. A second test is related to the length of the summary. It should contain fewer sentences than the original text. The summary length in terms of the number of sentences is fixed and known before summary generation.

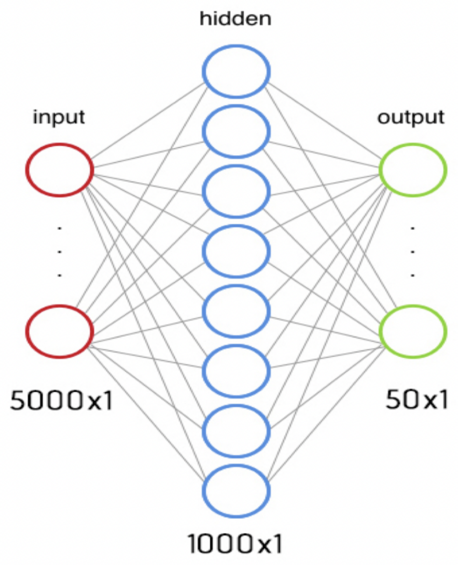


Figure 1: Neural Network

3 DESIGN AND IMPLEMENTATION

This project focuses on generating a highlighted version of a document based on the most relevant ideas contained. The main point would be the generation of a new output that could potentially be useful for the average user. Moreover, this study focuses on extraction of sentences from the text instead of any further analysis that could match the words in a query with the expected output. In this section, I will discuss the data set used in my research, the model to implement, and finally, the evaluation techniques for the capstone project.

3.1 Research Data

TREC (Text Research Collection Volume) [6] has been used across several papers that deal with text summaries. TREC is an effort to advance the state of the art in effective document detection (information retrieval) and data extraction from large, real-world data collections. The content is divided between five different disks. TREC is a test collection used for information retrieval research. It includes material from various sources such as the *Wall Street Journal, AP Newswire, Federal Register, San Jose Mercury News, U.S. Patents*, and computer-related articles. The collection is divided into several disks based on the source and year of the material. Neto et al. used the TREC Collection in a NN approach [12]. In their paper they performed two series of experiments, first implementing a Neural Network and second implementing statistical grouping of words. For both cases they used the same source of data from the TREC project, testing their approaches using material from the Wall Street Journal [9]. Later in the paper they go into detail about other possible sources of information they plan to use to further test their proposed approaches, one of them being magazines about computers.

This research used the disks four and five as training data. The collections contain documents that include material from the Financial Times Limited, the Congressional Record of the 103rd Congress, the Federal Register, and the Los Angeles Times. The first set of training documents came from the Los Angeles Times.

3.2 Pre-Processing

Before implementing any model to tokenize the text, it is essential to preprocess a document before using it in any NLP task because it helps to transform the raw, unstructured text into a structured format that NLP algorithms can quickly analyze. Preprocessing involves various steps such as tokenization and stop word removal, which helps reduce the data's size and eliminate noise and irrelevant information. These steps improve the data quality and ensure that the NLP algorithms can focus on the relevant information, enhancing their accuracy and efficiency.

In this case, the documents contained in each paper were crowded with different syntax notations as HTML code or side notes that would decrease the learning process of any NLP process. I started by using a simple Python script and removing any lines that began with "<. " This would remove every HTML reference and rewrite a new document with only essential information. Furthermore, I used the Gensim library to tackle the rest of the data preprocessing. Gensim is an open-source Python library for natural language processing (NLP) tasks. It provides a set of algorithms and tools for tasks such as topic modeling, similarity detection, and text summarization. Gensim is designed to handle large-scale, sparse, and unstructured text data and provides efficient algorithms for modeling and processing text data. Gensim is widely used in academic

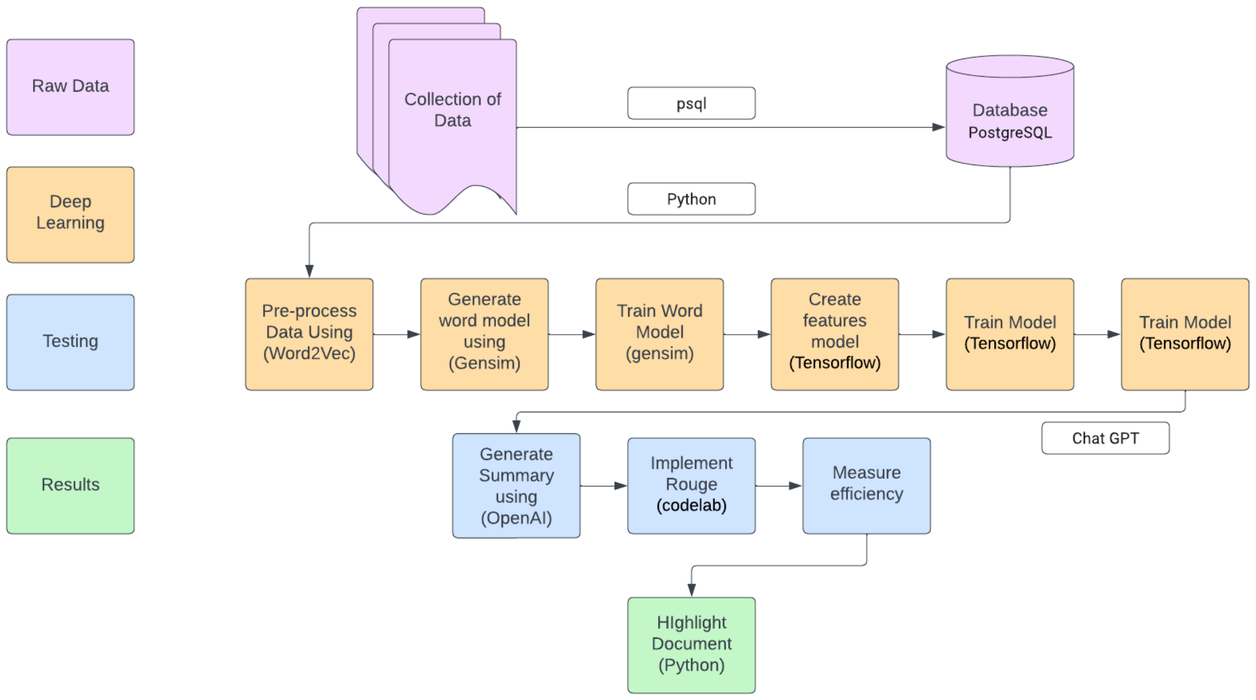

Raw Data
Deep Learning
Testing
Results
Collection of Data
psql
Database PostgreSQL
Python
Pre-process Data Using (Word2Vec)
Generate word model using (Gensim)
Train Word Model (gensim)
Create features model (Tensorflow)
Train Model (Tensorflow)
Train Model (Tensorflow)
Chat GPT
Generate Summary using (OpenAI)
Implement Rouge (codelab)
Measure efficiency
HIghlight Document (Python)

Figure 2: Software Architecture

and industrial research for various NLP tasks such as sentiment analysis, document clustering, and information retrieval [14].

The function used was $simple_preprocess()$ it takes a document (a string of text) as input and returns a list of tokens, where each token is a processed word from the original text. The resulting tokens can then be used for further NLP tasks such as topic modeling, sentiment analysis, and more. The analysis is done by the cosine similarity equation:

$$\cos\theta = \frac{x \cdot y}{\|x\| \, \|y\|}$$

3.3 Summarization Model

This section will detail a step-by-step guide to produce a reliable model that outputs an extractive summary. Extractive summarization is essential, selecting important or representative sentences from a text and assembling them into a shorter form. It involves identifying the most relevant information and preserving the meaning and tone of the original text.

3.3.1 Word2vec. Word2vec is a technique for creating word embeddings and vector representations of words in a high-dimensional space. The method uses a neural network to generate these embeddings by training on a large corpus of text data to predict the likelihood of a given word occurring in the context of other words in the same sentence. The resulting word embeddings help capture the semantic relationships between words in the text.

After the preprocessing stage, the documents are ready to generate the first model using the gensim library. At the start of the training phase, I created two matrices – an Embedding matrix and a Context matrix. These two matrices have an embedding for each word in the vocabulary. Subsequently, the similarity is created by taking the dot product of the input embedding with each of the context embeddings. Then, the Sigmoid function fits those values between 0 and 1, or in other words, generates the probability value:

$$f(x) = \frac{1}{1 + e^{-x}}$$

Finally, we create the function to train a model with the predefined epochs. The embeddings continue to be improved while we cycle through our entire dataset for some epochs. The output is a model where all the words are tokenized and ready for further analysis.

3.3.2 Features model. This process will replicate the methods used by Kaiaikhah et al.[7] by implementing different features and manipulating the hidden layers to output hierarchized sentences of the text in question. There will be an implementation of four features that encapsulate different criteria in the text.

Feature	Description
Feature 1	Tittle words in text
Feature 2	Name entities
Feature 3	Number of Cue Words
Feature 4	topic sentences

3.3.3 Feature 1. is designed to extract the words that match the tittle. First, it imports necessary libraries such as os, glob, gensim, keras, and numpy. Then define the *gettitles* function that reads all files in a specified directory, extracts the headlines, and returns a list of preprocessed and pruned titles. The *preprocessfilecontent* function is used to clean up the file content by removing headlines and returning the processed content.

The *matchtitles* function takes a directory path and a list of titles as input, tokenizes the titles, and iterates through the files to create a dictionary (*filetensor*) with keys as "title0", "title1", and so on. For each line in the preprocessed content, the function checks if any word from the current title appears in the line. If yes, it creates a feature array where each element represents the presence of a word from the tokenizer's word index in the line, and appends this array to the corresponding key in *filetensor*. The function then increments the title index if any word from the next title appears in the line.

Finally, the code creates a *titlewordfeatures* array by concatenating feature arrays for all articles. This array can be used as an input for a neural network to analyze or classify the articles based on the presence of words from the headlines in their content.

3.3.4 Feature 2. is designed to extract named entities and sentences containing named entities from a collection of text files. It begins by importing the necessary libraries, such as glob, os, re, spacy, numpy, defaultdict, *preprocessstring*, and Dictionary. The spacy library is loaded with the *encorewebsmmodel* to process text. To accomplish this, I defined three functions: *removehtmltags*, which takes a text input and removes any HTML tags using regular expressions. *preprocess*, this function preprocesses text using the *preprocessstring* function from the gensim library. And $extractinfofrom_file$ that takes a file path, reads the file line by line, and extracts named entities and sentences containing named entities. It uses the spacy library to identify named entities and excludes entities labeled as 'CARDINAL'. It then preprocesses the sentences, creates a dictionary of unique words, and converts the preprocessed sentences to a bag-of-words representation. Finally, it converts the bag-of-words sentences to a list of word frequencies and stores the results in a dictionary with the document ID as the key. The main function, *extractinfofromdirectory*, takes a directory path as input and iterates through all files in the directory with the specified pattern. It calls the *extractinfofromfile* function for each file and appends the resulting word frequency lists to a master list (results). After processing all the files, the list of lists is converted to a numpy array with dtype object. This numpy array, named *namedentityfeatures*, can be used for further analysis or as input for machine learning models.

3.3.5 Feature 3. The main objective is to process a collection of text files, identify the presence of specific cue words in each sentence, and tokenize the sentences. First, the code imports necessary libraries such as `os`, `glob`, `gensim`, `re`, `Tokenizer`, and *padsequences*. A list of cue words is defined, which consists of common transition words and phrases. The path to the data directory is set using the `os.path.expanduser` function. A `Tokenizer` object is instantiated for later use.

Two functions are defined in the code:

preprocessfilecontent: This function processes the file content, extracting headlines and appending a unique article identifier before each headline. *matchcuewords*: This function takes the path to the files and the list of cue words as input. For each file in the specified path, the function tokenizes sentences and checks for the presence of cue words. It then creates a dictionary, *filetensor*, with keys representing individual articles and values containing tokenized sentences and their corresponding cue word features. The sentences are padded using the *padsequences* function to ensure all sentences have the same length. The code concludes by returning the *filetensor* dictionary and the maximum sentence length.

3.3.6 Feature 4. The goal is to extract topic sentences from a collection of text files and tokenize them. First, the required libraries are imported, including `PlaintextParser`, `Tokenizer`, LexRankSummarizer, and *simplepreprocess*. The path to the data directory is set using the `os.path.expanduser` function.

Two functions are defined in the code:

gettopicsentences: This function takes the path to the files as input, iterates through each file, and extracts articles from the content. It then uses the LexRank algorithm from the `sumy` library to extract topic sentences from each paragraph of every article. The extracted topic sentences are stored in a dictionary with keys representing the article number and values containing the list of topic sentences. *tokenizetopicsentences*: This function takes the topic sentences as input and tokenizes them using the `gensim` library's *simplepreprocess* function. The tokenized sentences are stored in a dictionary with keys representing the article number and values containing the list of tokenized topic sentences. After defining these functions, the code calls *gettopicxsentences* to extract topic sentences from the articles and stores them in the *topicsentences* dictionary. Next, the *tokenizetopicsentences* function is called to tokenize the extracted topic sentences, which are then stored in the *tokenizedtopicsentences* dictionary. Finally, the tokenized topic sentences are converted to a list of lists named *topicsentencelist* for further analysis or use as input for machine learning models.

3.3.7 Highlighted Version. Finally, I used the selected sentences and words from the summary to highlight the Python document. For this, I implemented the termcolor library, a Python library for formatting text output in a terminal or console window. The first step was to open the input document and read its content into a string variable. Next, I added the summarized sentences to highlight in the document. Then, I processed the text to find the locations of these words in the document. Once the positions of the words had been identified, I used the termcolor library or any other text formatting library in Python to highlight the words in the document. Finally, I wrote the highlighted text into a new file, to overwrite the original file with the highlighted version.

4 EVALUATION

A significant portion of the NN-based summaries' testing is associated with the accuracy of a generated summary, which is often evaluated by comparing it to already annotated versions of documents and by extracting relevant items. For instance, Kaiaikhah [15] employed 25 different news articles, which were independently summarized by a human reader and all three modified networks.

The average accuracy of the discretized real-values into intervals neural network was 96 percent.

To examine the evaluation process more thoroughly, we incorporated OpenAI's state-of-the-art language model as an additional input for comparing the rest of our neural network. We used the widely accepted ROUGE scores to measure the performance of our model, thus avoiding human involvement. ROUGE stands for Recall-Oriented Understudy for Gisting Evaluation, and it includes measures that automatically determine the quality of a summary by comparing it to other (ideal) summaries created by humans. These measures count the number of overlapping units, such as n-grams, word sequences, and word pairs, between the computer-generated summary being evaluated and the ideal summaries crafted by humans [9]. By leveraging OpenAI's model and ROUGE scores based on n-grams, we conducted a comprehensive assessment of our neural network's performance in generating accurate summaries.

Table 1: NN ROUGE Scores

	Precision	Recall	F1 Score
ROUGE-1	0.60	0.53	0.58
ROUGE-2	0.25	0.29	0.27
ROUGE-L	0.83	0.72	0.79

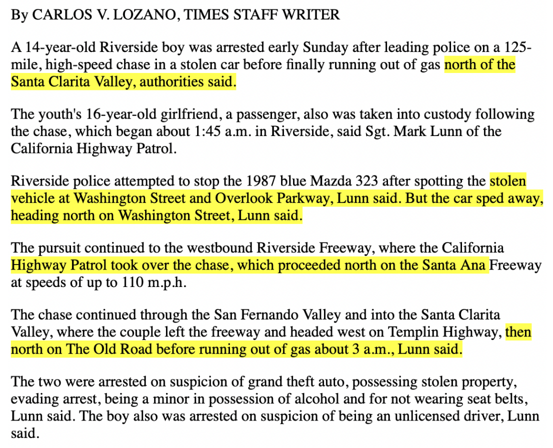
The table presents the ROUGE scores for a neural network-generated summary compared to OpenAI-generated summaries. The scores are broken down into Precision, Recall, and F1 Score for each of the three ROUGE metrics (ROUGE-1, ROUGE-2, and ROUGE-L).

(1) ROUGE-1: This score measures the overlap of individual words (unigrams) between the summaries. The precision is 0.60, meaning that 60 percent of the words in the neural network-generated summary also appear in the OpenAI-generated summaries. The recall is 0.53, indicating that 53 percent of the words in the OpenAI-generated summaries are present in the neural network-generated summary. The F1 Score, which balances both precision and recall, is 0.58, suggesting a moderate level of word overlap between the summaries.
(2) ROUGE-2: This score measures the overlap of bigrams (two consecutive words) between the summaries. The precision is 0.25, and the recall is 0.29, both lower than the ROUGE-1 scores. This indicates that the neural network-generated summary has a lower level of coherence and structural similarity to the OpenAI-generated summaries. The F1 Score for ROUGE-2 is 0.27, confirming the lower overlap in terms of bigrams.
(3) ROUGE-L: This score, based on the Longest Common Subsequence (LCS), measures the overall similarity between the summaries while allowing for flexibility in word order. The precision is 0.83, and the recall is 0.72, both relatively high values, suggesting that the neural network-generated summary and the OpenAI-generated summaries share a substantial amount of information, despite potential differences in phrasing or word order. The F1 score for ROUGE-L is 0.79, indicating a good level of similarity between the summaries.

In summary, the ROUGE scores suggest that the neural network-generated summary shares a moderate level of word overlap (ROUGE-1) and a good level of overall similarity (ROUGE-L) with the OpenAI-generated summaries. However, the lower ROUGE-2 scores indicate that there may be differences in coherence and structure between the summaries, which makes sense based on the fact that the Neural Network is extracting textual information from the text, without considering cohesion.

5 RESULTS

In this section we will explore the results for each of the individual features and the result overall of the text.



By CARLOS V. LOZANO, TIMES STAFF WRITER

A 14-year-old Riverside boy was arrested early Sunday after leading police on a 125-mile, high-speed chase in a stolen car before finally running out of gas north of the Santa Clarita Valley, authorities said.

The youth's 16-year-old girlfriend, a passenger, also was taken into custody following the chase, which began about 1:45 a.m. in Riverside, said Sgt. Mark Lunn of the California Highway Patrol.

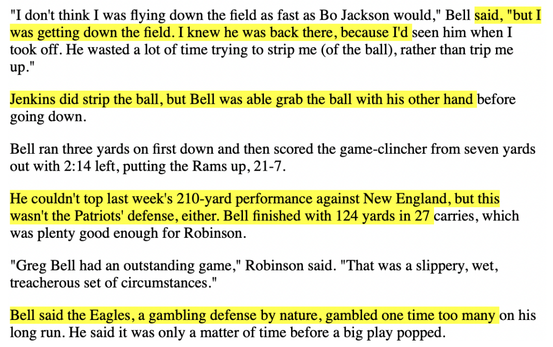
Riverside police attempted to stop the 1987 blue Mazda 323 after spotting the stolen vehicle at Washington Street and Overlook Parkway, Lunn said. But the car sped away, heading north on Washington Street, Lunn said.

The pursuit continued to the westbound Riverside Freeway, where the California Highway Patrol took over the chase, which proceeded north on the Santa Ana Freeway at speeds of up to 110 m.p.h.

The chase continued through the San Fernando Valley and into the Santa Clarita Valley, where the couple left the freeway and headed west on Templin Highway, then north on The Old Road before running out of gas about 3 a.m., Lunn said.

The two were arrested on suspicion of grand theft auto, possessing stolen property, evading arrest, being a minor in possession of alcohol and for not wearing seat belts, Lunn said. The boy also was arrested on suspicion of being an unlicensed driver, Lunn said.

Figure 3: Title Words



"I don't think I was flying down the field as fast as Bo Jackson would," Bell said, "but I was getting down the field. I knew he was back there, because I'd seen him when I took off. He wasted a lot of time trying to strip me (of the ball), rather than trip me up."

Jenkins did strip the ball, but Bell was able grab the ball with his other hand before going down.

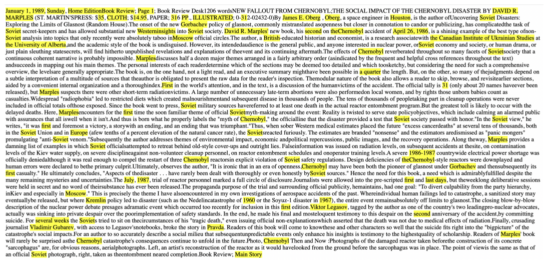
Bell ran three yards on first down and then scored the game-clincher from seven yards out with 2:14 left, putting the Rams up, 21-7.

He couldn't top last week's 210-yard performance against New England, but this wasn't the Patriots' defense, either. Bell finished with 124 yards in 27 carries, which was plenty good enough for Robinson.

"Greg Bell had an outstanding game," Robinson said. "That was a slippery, wet, treacherous set of circumstances."

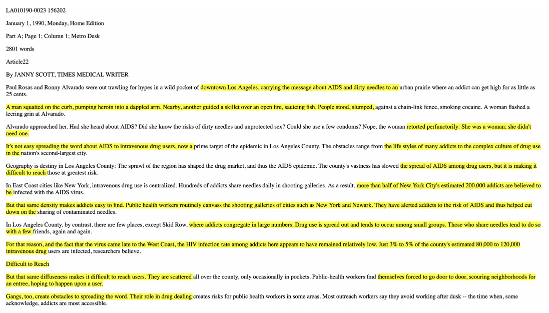
Bell said the Eagles, a gambling defense by nature, gambled one time too many on his long run. He said it was only a matter of time before a big play popped.

Figure 4: Topic Sentence



January 1, 1989, Sunday, Home EditionBook Review; Page 1; Book Review Desk1206 wordsNEW FALLOUT FROM CHERNOBYL;THE SOCIAL IMPACT OF THE CHERNOBYL DISASTER BY DAVID R. MARPLES (ST. MARTIN'SPRESS: $35, CLOTH; $14.95, PAPER; 316 PP., ILLUSTRATED; 0-312-02432-0)By James E. Oberg , Oberg, a space engineer in Houston, is the author ofUncovering Soviet Disasters: Exploring the Limits of Glasnost (Random House).The onset of the new Gorbachev policy of glasnost, commonly mistranslated asopenness but closer in connotation to candor or publicizing, has complicatedthe task of Soviet secret-keepers and has allowed substantial new Westerninsights into Soviet society. David R. Marples' new book, his second on theChernobyl accident of April 26, 1986, is a shining example of the best type ofnon-Soviet analysis into topics that only recently were absolutely taboo inMoscow official circles.The author, a British-educated historian and economist, is a research associatewith the Canadian Institute of Ukrainian Studies at the University of Alberta,and the academic style of the book is undisguised. However, its intendedaudience is the general public, and anyone interested in nuclear power, orSoviet economy and society, or human drama, or just plain sleuthing statesecrets, will find hitherto unpublished revelations and explanations of theevent and its continuing aftermath.The effects of Chernobyl reverberated throughout so many facets of Sovietsociety that a continuous coherent narrative is probably impossible. Marplesdiscusses half a dozen major themes arranged in a fairly arbitrary order (asindicated by the frequent and helpful cross references throughout the text) andsucceeds in mapping out his main themes. The personal interests of each readerdetermine which of the sections may be deemed too detailed and which toosketchy, but considering the need for such a comprehensive overview, the levelsare generally appropriate.The book is, on the one hand, not a light read, and an executive summary mighthave been possible in a quarter the length. But, on the other, so many of thejudgments depend on a subtle interpretation of a multitude of sources that theauthor is obligated to present the raw data for the reader's inspection. Themodular nature of the book also allows a reader to skip, browse, and revisitearlier sections, aided by a convenient internal organization and a thoroughindex.First in the world's attention, and in the text, is a discussion of the humanvictims of the accident. The official tally is 31 (only about 20 names haveever been released), but Marples suspects there were other short-term radiationvictims. A large number of unnecessary late-term abortions were also performedon local women, and by rights those unborn babies count as casualties.Widespread "radiophobia" led to restricted diets which created malnourishmentand subsequent disease in thousands of people. The tens of thousands of peopletaking part in cleanup operations were never included in official totals ofthose exposed. Since the book went to press, Soviet military sources havereferred to at least one death in the actual reactor entombment program.But the greatest toll is likely to occur with the delayed deaths. Here, Marplesencounters for the first time the soon familiar theme of official Sovietmyth-making around the event: Reality is twisted to serve state policyobjectives, which include calming an alarmed public with assurances that all iswell when it isn't.And thus is born what he properly labels the "myth of Chernobyl," the officialline that the disaster provided a test that Soviet society passed with honor."In the Soviet view," he writes, "it was first and foremost a victory, a storywith an ending, and an ending that was triumphant."Thus, when sober Western medical estimates placed the future "excess cancerdeaths" at several tens of thousands, both in the Soviet Union and in Europe (afew tenths of a percent elevation of the natural cancer rate), the Sovietsreacted furiously. The estimates are branded "nonsense" and the estimators aredismissed as "panic mongers" promulgating "anti-Soviet venom."Subsequently the author addresses themes of environmental impact, economic andpolitical repercussions, public images, and the recovery operations. Along theway, Marples provides a damning list of examples in which Soviet officialsattempted to retreat behind old-style cover-ups and outright lies. Falseinformation was issued on radiation levels, on subsequent accidents at thesite, on contamination levels of the Kiev water supply, on severe disciplineagainst non-volunteer cleanup personnel, on reactor entombment schedules and onoperator training levels.A severe 1986-1987 countrywide electrical power shortage was officially deniedalthough it was real enough to compel the restart of three Chernobyl reactorsin explicit violation of Soviet safety regulations. Design deficiencies of theChernobyl-style reactors were downplayed and human errors were declared to bethe primary culprit.Ultimately, observes the author, "It is ironic that in an era of openness,Chernobyl may have been both the pioneer of glasnost under Gorbachev and thensubsequently its first casualty." He ultimately concludes, "Aspects of thedisaster . . . have rarely been dealt with thoroughly or even honestly bySoviet sources." Hence the need for this book, a need which is admirablyfulfilled despite the many remaining mysteries and uncertainties.The July, 1987, trial of reactor personnel marked a full circle of disclosure.Journalists were allowed into the pre-scripted first and last days, but theweeklong deliberative sessions were held in secret and no word of theirsubstance has ever been released.The propaganda purpose of the trial and surrounding official publicity, hemaintains, had one goal: "To divert culpability from the party hierarchy, inKiev and especially in Moscow." This is precisely the theme I have alsoencountered in my own investigations of aerospace accidents of the past. Whereindividual human failings led to catastrophe, a sanitized story may eventuallybe released, but where Kremlin policy led to disaster (such as the Nedelincatastrophe of 1960 or the Soyuz-1 disaster in 1967), the entire event remainsabsolutely off limits to glasnost.The closing blow-by-blow description of the nuclear power debate presages adramatic event which occurred too recently for inclusion in this first edition.Viktor Legasov, tagged by the author as one of the country's two leadingpro-nuclear advocates, actually was sinking into private despair over the poorimplementation of safety standards. In the end, he made his final and mosteloquent testimony to this despair on the second anniversary of the accident,by committing suicide. For several weeks the Soviets tried to sit on thecircumstances of his "tragic death," even issuing official non-explanationswhich asserted that the death was not due to medical effects of radiation.Finally, crusading journalist Vladimir Gubarev, with access to Legasov'snotebooks, broke the story in Pravda. Readers of this book will come to knowthese and other characters so well that the suicide fits right into the "bigpicture" of the catastrophe's social impacts.For an author to so accurately describe a social milieu that subsequentunpredictable events only enhance his insights is testimony to the highestquality of scholarship. Readers of Marples' book will rarely be surprised asthe Chernobyl catastrophe's consequences continue to unfold in the future.Photo, Chernobyl Then and Now :Photographs of the damaged reactor taken beforethe construction of its concrete "sarcophagus" are, for obvious reasons, aerialphotographs. Left, an artist's reconstruction of the reactor as it would havelooked from the ground before the sarcophagus was in place. The point of viewis the same as that of an official Soviet photograph, right, taken as theentombment neared completion.Book Review; Main Story

Figure 5: Named Entities



LA010190-0023 156202

January 1, 1990, Monday, Home Edition

Part A; Page 1; Column 1; Metro Desk

2801 words

Article22

By JANNY SCOTT, TIMES MEDICAL WRITER

Paul Rosas and Ronny Alvarado were out trawling for hypes in a wild pocket of downtown Los Angeles, carrying the message about AIDS and dirty needles to an urban prairie where an addict can get high for as little as 25 cents.

A man squatted on the curb, pumping heroin into a dappled arm. Nearby, another guided a skillet over an open fire, sauteing fish. People stood, slumped, against a chain-link fence, smoking cocaine. A woman flashed a leering grin at Alvarado.

Alvarado approached her. Had she heard about AIDS? Did she know the risks of dirty needles and unprotected sex? Could she use a few condoms? Nope, the woman retorted perfunctorily: She was a woman; she didn't need one.

It's not easy spreading the word about AIDS to intravenous drug users, now a prime target of the epidemic in Los Angeles County. The obstacles range from the life styles of many addicts to the complex culture of drug use in the nation's second-largest city.

Geography is destiny in Los Angeles County: The sprawl of the region has shaped the drug market, and thus the AIDS epidemic. The county's vastness has slowed the spread of AIDS among drug users, but it is making it difficult to reach those at greatest risk.

In East Coast cities like New York, intravenous drug use is centralized. Hundreds of addicts share needles daily in shooting galleries. As a result, more than half of New York City's estimated 200,000 addicts are believed to be infected with the AIDS virus.

But that same density makes addicts easy to find. Public health workers routinely canvass the shooting galleries of cities such as New York and Newark. They have alerted addicts to the risk of AIDS and thus helped cut down on the sharing of contaminated needles.

In Los Angeles County, by contrast, there are few places, except Skid Row, where addicts congregate in large numbers. Drug use is spread out and tends to occur among small groups. Those who share needles tend to do so with a few friends, again and again.

For that reason, and the fact that the virus came late to the West Coast, the HIV infection rate among addicts here appears to have remained relatively low. Just 3% to 5% of the county's estimated 80,000 to 120,000 intravenous drug users are infected, researchers believe.

Difficult to Reach

But that same diffuseness makes it difficult to reach users. They are scattered all over the county, only occasionally in pockets. Public-health workers find themselves forced to go door to door, scouring neighborhoods for an entree, hoping to happen upon a user.

Gangs, too, create obstacles to spreading the word. Their role in drug dealing creates risks for public health workers in some areas. Most outreach workers say they avoid working after dusk -- the time when, some acknowledge, addicts are most accessible.

Figure 6: Cue Words

6 CONCLUSION

This study demonstrates the potential of utilizing a neural network-based approach to generate summaries from text documents. By implementing and combining various features, including title words, named entities, cue words, and topic sentences, the model is able to extract significant information from the source text and generate a summary. The evaluation of the neural network's performance using ROUGE scores shows promising results, with moderate to good similarity levels compared to summaries generated by OpenAI's state-of-the-art language model.

However, there is still room for improvement in the coherence and structural similarity between the neural network-generated summaries and the ideal human-generated summaries. This can potentially be addressed by incorporating more sophisticated techniques, such as attention mechanisms or transformer models, which have shown excellent performance in natural language understanding and generation tasks. Additionally, incorporating more advanced features or exploring ways to improve the interaction between features could also lead to better summary quality.

In conclusion, the neural network-based summarization approach presented in this study is a promising starting point for generating accurate summaries. With further refinement and the incorporation of advanced techniques, the performance of the model can be improved to generate even more coherent and informative summaries.

7 ACKNOWLEDGEMENTS

I would like to thank David Barbella and Sofia Lemons for providing me detailed feedback to improve this topic proposal paper.

REFERENCES

[1] Breck Baldwin and Thomas S. Morton. 1998. Dynamic coreference-based summarization. In *Proceedings of the third conference on empirical methods for natural language processing*. 1–6.

[2] Adam Berger and Vibhu O Mittal. 2000. Query-relevant summarization using FAQs. In *Proceedings of the 38th annual meeting of the association for computational linguistics*. 294–301.

[3] Saeedeh Gholamrezazadeh, Mohsen Amini Salehi, and Bahareh Gholamzadeh. 2009. A comprehensive survey on text summarization systems. In *2009 2nd International Conference on Computer Science and its Applications*. IEEE, 1–6.

[4] Yoav Goldberg and Omer Levy. 2014. word2vec Explained: deriving Mikolov et al.'s negative-sampling word-embedding method. *arXiv preprint arXiv:1402.3722* (2014).

[5] Ed Gordon. 1996. Verity agent technology: Automatic filtering, matching and dissemination of information. *Vine* (1996).

[6] Donna Harman and Mark Liberman. 2012. Tipster complete. https://catalog.ldc.upenn.edu/LDC93T3A

[7] Khosrow Kaikhah. 2004. Automatic text summarization with neural networks. In *2004 2nd International IEEE Conference on'Intelligent Systems'. Proceedings (IEEE Cat. No. 04EX791)*, Vol. 1. IEEE, 40–44.

[8] Danny H Lie. 1998. Sumatra: a system for automatic summary generation. *Carp Technologies* (1998).

[9] Chin-Yew Lin. 2004. ROUGE: A Package for Automatic Evaluation of Summaries. In *Text Summarization Branches Out*. Association for Computational Linguistics, Barcelona, Spain, 74–81. https://aclanthology.org/W04-1013

[10] Inderjeet Mani. 2001. *Automatic summarization*. Vol. 3. John Benjamins Publishing.

[11] Nikita Munot and Sharvari S Govilkar. 2014. Comparative study of text summarization methods. *International Journal of Computer Applications* 102, 12 (2014).

[12] Joel Larocca Neto, Alex A Freitas, and Celso AA Kaestner. 2002. Automatic text summarization using a machine learning approach. In *Brazilian symposium on artificial intelligence*. Springer, 205–215.

[13] Phil Picton. 1994. What is a neural network? In *Introduction to Neural Networks*. Springer, 1–12.

[14] Radim Rehurek and Petr Sojka. 2011. Gensim–python framework for vector space modelling. *NLP Centre, Faculty of Informatics, Masaryk University, Brno, Czech Republic* 3, 2 (2011).

[15] Aakash Sinha, Abhishek Yadav, and Akshay Gahlot. 2018. Extractive text summarization using neural networks. *arXiv preprint arXiv:1802.10137* (2018).

[16] Yong SP, Ahmad IZ Abidin, and YY Chen. 2006. A neural-based text summarization system. *WIT Transactions on Information and Communication Technologies* 37 (2006).

[17] Sasha Spala, Franck Dernoncourt, Walter Chang, and Carl Dockhorn. 2018. A Comparison Study of Human Evaluated Automated Highlighting Systems. In *Proceedings of the 32nd Pacific Asia Conference on Language, Information and Computation*.

[18] Sasha Spala, Franck Dernoncourt, Walter Chang, and Carl Dockhorn. 2018. A Web-based Framework for Collecting and Assessing Highlighted Sentences in a Document. In *Proceedings of the 27th International Conference on Computational Linguistics: System Demonstrations*. Association for Computational Linguistics, Santa Fe, New Mexico, 78–81. https://aclanthology.org/C18-2017

[19] Peter D. Turney. 2002. Learning to Extract Keyphrases from Text. https://doi.org/10.48550/ARXIV.CS/0212013

[20] Sun-Chong Wang. 2003. Artificial neural network. In *Interdisciplinary computing in java programming*. Springer, 81–100.