Adv Mobile Learn Educ Res, 2025, 5(1): 1344-1355
DOI: 10.25082/AMLER.2025.01.010

RESEARCH ARTICLE

Automated Video Title Generation for Mobile Learning Resources: A Deep Learning Approach with Educational Context Awareness

Zheng Gong[1,2]

[1] Department of Applied Mathematics, Hong Kong Polytechnic University, Hong Kong, China
[2] School of Computer Science & Technology, Beijing Institute of Technology, Beijing, China

Correspondence to: Zheng Gong, Department of Applied Mathematics, Hong Kong Polytechnic University, Hong Kong, China; Email: 1390465790@qq.com

Received: December 3, 2024;
Accepted: March 8, 2025;
Published: March 13, 2025.

Citation: Gong, Z. (2025). Automated Video Title Generation for Mobile Learning Resources: A Deep Learning Approach with Educational Context Awareness. *Advances in Mobile Learning Educational Research*, 5(1), 1344-1355.
https://doi.org/10.25082/AMLER.2025.01.010

Abstract: With the rapid growth of mobile learning platforms, short educational videos have emerged as a critical resource for learners. However, manually generating concise and pedagogically meaningful titles for these videos remains a time-consuming challenge. To address this issue, this study proposes a deep learning framework designed for automated video title generation in educational contexts. The framework integrates Convolutional Neural Networks (CNNs), Long Short-Term Memory (LSTM) networks, and natural language processing (NLP) techniques, with explicit awareness of pedagogical relevance. The proposed approach operates in three stages: 1) extracting key frames from input videos using an optimized shot detection algorithm, 2) analyzing these frames with CNN models to derive semantic representations of visual content, and 3) processing the representations through an LSTM network to generate descriptive text. The output is further refined using the TextRank algorithm to ensure conciseness and contextual coherence. Experimental results demonstrate that our framework effectively generates high-quality video titles that are both educationally informative and contextually engaging, outperforming baseline methods in alignment with curriculum standards and learner-centric search intent.

Keywords: mobile learning, deep learning, convolutional neural networks, natural language processing, video captioning

1 Introduction

1.1 Background and Motivation

The rapid expansion of short video platforms has led to an exponential increase in video content. According to the "China Online Audio-Visual Development Research Report (2024)", as of December 2023, China had over 1.074 billion online audio-visual users, with short videos accounting for the highest user engagement. The immense volume of video uploads necessitates efficient methods for categorization, recommendation, and compliance verification. Video titles play a pivotal role in these processes by summarizing content and attracting viewer attention (Zhang et al., 2014; Jiang & Zhang, 2003). However, manually crafting titles for the overwhelming influx of videos is both labor-intensive and inefficient. This challenge highlights the need for automated methods that leverage deep learning techniques to generate high-quality short video titles.

1.2 Related Work

Traditional educational video title generation methods relied on handcrafted features and predefined templates, which lacked adaptability and scalability (Uygun, 2024). With the advent of deep learning, encoder-decoder frameworks leveraging sequence-to-sequence learning have become dominant (Ladias et al., 2021). Recent studies (Venugopalan et al., 2015; Wilson et al., 2023; Shu et al., 2019) have explored the use of CNNs for visual feature extraction and recurrent neural networks (RNNs), particularly LSTM architectures, for text generation. Additionally, NLP algorithms such as TextRank have been employed to refine generated text by identifying the most relevant phrases. However, challenges remain in optimizing feature fusion, handling multimodal data, and ensuring linguistic coherence in generated titles.

1.3 Research Objectives

This study aims to develop an automated video title generation system that:

(1) Video Reading and Keyframe Extraction: Utilize OpenCV (Suarez et al., 2014) to read video files and decompose the video content into individual shots through shot segmentation. Subsequently, within each shot, content keyframes capturing the core information of the shot are further extracted.

(2) Image Caption Generation: Combine Convolutional Neural Networks (CNN) and Long Short-Term Memory networks (LSTM) (Neil et al., 2016) to address image annotation and sentence retrieval tasks. First, high-level semantic features are extracted from input images using CNN. These features are then fed into an LSTM to iteratively predict the next most probable word, ultimately forming a descriptive text for the image.

(3) Text Summarization: Generate a text summary of the video content using the TextRank algorithm (Mihalcea & Tarau, 2004). Sentences are converted into term frequency vectors, and the cosine similarity between these vectors is calculated to determine sentence similarity (Goldberg & Levy, 2014). The importance of each sentence in the text is then evaluated based on its contextual relevance.

(4) Post-processing with NLP: Apply natural language processing (NLP) techniques to refine the generated text summary. This includes rewriting the summary into a concise and engaging short video title that effectively highlights the video's content.

2 Frameworks and Tools

2.1 Deep Learning Framework: PyTorch

PyTorch (Imambi et al., 2021), an open-source deep learning framework, was utilized due to its dynamic computation graph and ease of debugging. Its flexible architecture enables efficient model training and inference on GPUs. PyTorch has become a leading deep learning framework due to its exceptional computational performance, robust functionality, and broad applicability. These attributes have garnered it substantial acclaim and widespread adoption across both academic and industrial domains.

2.2 Computer Vision Library: OpenCV

OpenCV (Bradski & Kaehler, 2000) facilitates video processing tasks, including frame extraction, object detection, and image enhancement. Its optimized performance is well-suited for real-time applications. OpenCV is architected with a principal emphasis on computational efficiency, a design imperative that has solidified its dominance in real-time vision systems. The framework's inherent scalability across multi-core architectures, coupled with its streamlined API, positions it as a foundational tool for deploying performant and user-friendly computer vision solutions in both academic and industrial settings.

2.3 NLP Toolkit: HanLP

HanLP (Liu et al., 2021) is a multi-language NLP toolkit supporting tokenization, part-of-speech tagging, dependency parsing, and keyword extraction. It was used to preprocess and refine generated text. HanLP's architecture is grounded in the integration of the most extensive multilingual textual resources globally, enabling unified processing of 10 core NLP tasks spanning 104 languages – from morphologically rich languages like Russian to logographic systems such as Chinese. Its unified pipeline systematically addresses tokenization, POS tagging, NER, syntactic parsing (dependency and constituency), semantic role labeling, and abstract meaning representation, establishing an empirically robust framework for cross-linguistic NLP research and industrial deployment.

3 Methodology

3.1 Key Frame Extraction

A three-frame difference method (Khosrovian et al., 2008) was employed to identify significant changes in consecutive frames, effectively capturing scene transitions and key moments (Figure 1). Let f_{n+1}, f_n, and f_{n-1} denote the images of the $(n+1)$-th, n-th, and $(n-1)$-th frames in a video sequence, respectively.

The grayscale values of the corresponding pixels in these three frames are denoted as

$f_{n+1}(x, y)$, $f_n(x, y)$, and $f_{n-1}(x, y)$. The difference images D_n and D_{n+1} are obtained as follows:

$$D'_n(x, y) = |f_{n+1}(x, y) - f_n(x, y)| \cap |f_n(x, y) - f_{n-1}(x, y)| \quad (1)$$

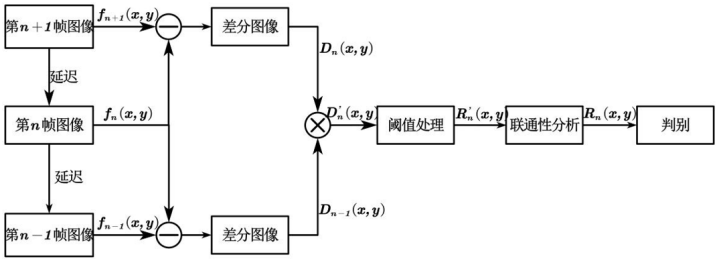

Figure 1 Key Frame Extraction Process

3.2 Feature Extraction Using CNN

A ResNet-based CNN model (Chang et al., 2016) was employed to extract high-level semantic features from key frames. The final fully connected layer was replaced with an identity mapping to retain essential feature representations. The high-level semantic features of input images are extracted through Convolutional Neural Networks (CNNNs), which are subsequently fed into Long Short-Term Memory (LSTM) networks to iteratively predict the next most probable words, thereby generating image captions. The loss function of the CNN is designed in a sequential manner to ensure linguistic alignment between the generated words and expected descriptions.

In our experimental framework, the ResNet model (Zhang & Woodland, 2015) serves as the foundational architecture for extracting image semantics. The residual structure of ResNet significantly accelerates neural network training convergence while substantially improving model accuracy. Furthermore, ResNet addresses critical limitations inherent in conventional CNN architectures, particularly the gradient explosion and degradation issues that emerge when stacking excessive numbers of deep network layers. (Figure 2)

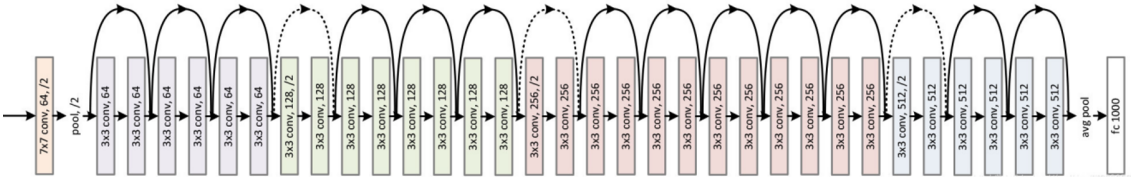

Figure 2 Schematic Diagram of the 34-Layer ResNet Model (Wang et al., 2012)

3.3 Text Generation Using LSTM

The extracted visual features were input to an LSTM network, which sequentially generated descriptive text. The LSTM model was trained on the AI Challenger dataset, which contains 300,000 images with corresponding textual descriptions. The experiments in this study focus on video-to-text summarization. To ensure conciseness and relevance, precision-oriented output control mechanisms are implemented, as not all LSTM-generated intermediate results are required for this task. (Figure 3)

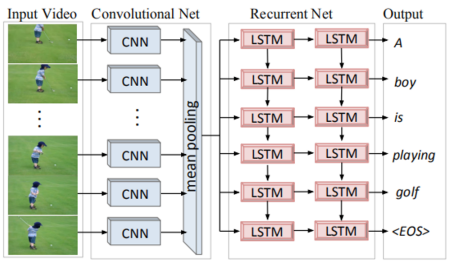

Figure 3 Workflow Diagram for Video Content Description Generation

3.4 Text Summarization Using TextRank

The generated text was further refined using the TextRank algorithm, which constructs a sentence similarity graph and ranks sentences based on importance scores. This step ensures

that the final video title is both concise and informative. In this study, two distinct methods were implemented for TextRank-based text processing.

(1) Term Frequency Vector Method

Sentences are converted into term frequency vectors, with cosine similarity between vectors serving as inter-sentence similarity (following textrank4zh's implementation). textrank4zh calculates keywords using N-grams + TextRank, where word adjacency determines edge connections between lexical nodes. This model assumes that physical proximity linearly correlates with semantic relevance while discounting long-range contextual dependencies. Such a premise may be suboptimal when distantly positioned words exhibit stronger semantic associations.

(2) Term Vector Averaging Method

By encoding semantic/syntactic relationships in multidimensional space through term frequency vectors, word2vec embeddings are applied for sentence similarity computation. (Figure 4 and 5)

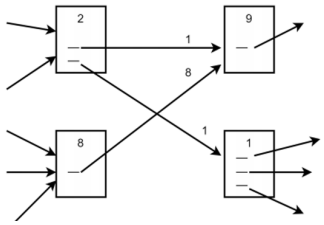

Figure 4 Schematic Diagram of the PageRank Algorithm (Langville & Meyer, 2004)

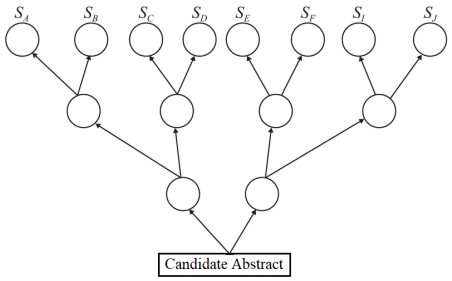

Figure 5 Candidate Abstract

For sentences A=['you','and','me'] and B=['good','word','python']: Generate word vectors: v1, v2, v3 for A → compute mean vector V_a(avg) = (v1 + v2 + v3)/3 Repeat for B to obtain V_b(avg) Calculate cosine similarity between V_a(avg) and V_b(avg) as the sentence similarity metric. This approach preserves contextual semantics while mitigating noise from redundant lexical features.

4 Results and Analysis

4.1 Key Frame Extraction Analysis

In this experiment, the video stream is read using OpenCV's cv2 library. The three-frame difference method is employed: 1) Compute first-order and second-order histogram difference metrics between consecutive frames. 2) Determine shot boundaries by comparing these metrics against predefined thresholds. 3) Record frame indices containing identified scene cuts. The implementation supports large video files through streaming processing, where data is incrementally loaded into memory to optimize resource utilization.

4.1.1 Implementation of Functions

In the *ShortDetect.py* script, a shot_detect class is defined, systematically modularizing the key frame extraction process and encapsulating it within dedicated functions. Furthermore, to optimize performance, the experiment implements distinct interfaces for short and long videos. The program dynamically selects the appropriate interface based on the video's duration, ensuring efficient memory allocation and enhancing processing speed. The functionalities of key functions are detailed showed in Table 1.

Table 1 ShortDetect.py function description

Function Name	Functional Description
get_normed_hist(self, frame)	Calculate the histogram and normalize, flatten it, and return a 1D array with the number of elements equal to hist_size * the number of channels.
merge_short_shots(self)	Calculate the number of frames between shots and the minimum number of frames between shots.
get_key_frames(video_path)	Extract key frames from the video file.
run_detect(self, video_path=None, batch_size=100)	Read the video file frame by frame in a streaming manner into memory. This interface is suitable for processing long videos.
run(self, video_path=None)	Load all frame data of the video file into memory. This interface is suitable for processing short videos.
test(in_dir, out_dir)	Used for testing programs and debugging.

Due to the wide variation in video resolutions, frames are standardized to 256 × 256 through preprocessing for efficient processing. Use the following code to implement the resizing:

```
cap = cv2.VideoCapture(self.video_path)
IMG_WIDTH = int(cap.get(cv2.CAP_PROP_FRAME_WIDTH)) # 视频帧宽
IMG_HEIGHT= int(cap.get(cv2.CAP_PROP_FRAME_HEIGHT)) # 视频帧高
if IMG_WIDTH > IMG_HEIGHT:
        IMG_SIZE = (256, int(256.0 * float(IMG_HEIGHT) / float(IMG_WIDTH)))
else:
        IMG_SIZE = (int(256.0 * float(IMG_WIDTH) / float(IMG_HEIGHT)), 256)
```

The core algorithm for video keyframe extraction is the three-frame difference method, which can be implemented in three steps:

(1) Calculate the histogram of a video frame, normalize and flatten it to generate a 1D array.

(2) Compute the difference image between adjacent frames by calculating the Manhattan distance between two 1D arrays.

(3) Determine shot transitions (keyframes) by calculating the first-order difference and second-order difference operators of the adjacent difference images, then apply threshold-based detection. Example code:

```
frame = cv2.resize(frame, IMG_SIZE, interpolation=cv2.INTER_CUBIC) # 视频帧预处理
if i == 1:  # 记录往前 2 帧
    frame_prev_2 = frame
elif i == 2:  # 记录往前 1 帧
    frame_prev_1 = frame
else:  # 从第 3 帧开始判断
    hist_prev_1 = self.get_normed_hist(frame_prev_1)
    hist_prev_2 = self.get_normed_hist(frame_prev_2)
    hist_cur = self.get_normed_hist(frame)
        # 计算两个 hist 的曼哈顿距离
    score_pre = cityblock(hist_prev_1, hist_prev_2)
    score_cur = cityblock(hist_cur, hist_prev_1)
        # 一阶,二阶差分阈值判断
    if (score_cur >= threshold) \
            and (abs(score_cur - score_pre) >= threshold * 0.5):
            key_frames.append(frame)  # 记录关键帧
        # 更新前两帧
    frame_prev_2 = frame_prev_1  # 更新往前 1 帧
    frame_prev_1 = frame  # 更新往前 2 帧
```

4.1.2 Functional Testing

Call the *test(in_dir, out_dir)* function from the *ShortDetect.py* script to test the keyframe extraction functionality. The test video is a 7-second clip excerpted from a specific NBA game recording. The test results showed in Figure 6.

```
ORIGIN_SIZE:  (720, 720)
IMG_SIZE:  (256, 256)
FRAME_NUM:  454
max diff: 2.4743853 min diff: 0.03925124
[(0, 84), (84, 253), (253, 334), (334, 449)]
```

Figure 6 Keyframe Extraction Test Results

The test results demonstrate that: 1) Original video resolution: 720 × 720, preprocessed to a standardized 2) 256 × 256. Total frames: 454. Second-order difference operator between adjacent frame histograms: Maximum value: 2.4743853, Minimum value: 0.03925124. This indicates at least one shot transition (i.e., a minimum of two keyframes) in the video. 3) The final output provides a merged video segment list (*e.g.*, (0, 84) denotes frames 0 to 84 as a continuous segment), where the interval endpoints correspond to keyframe IDs. Subsequent experimental steps will operate on these keyframes.

4.2 Visual Feature Extraction Performance

This study integrates deep convolutional neural networks (CNNs) and deep recurrent neural networks (RNNs) to address image captioning and sentence retrieval tasks. The CNN extracts high-level semantic features from input images, which are then fed into a Long Short-Term Memory (LSTM) network to iteratively predict the next most probable word, thereby generating image descriptions. The model is trained by progressively optimizing the neural network's loss function to align the generated words with the target descriptions.

4.2.1 Dataset

The training data for the image captioning model is sourced from the 2017 AI Challenger dataset (Wu et al., 2019), where each image is annotated with five Chinese descriptive sentences. The dataset contains 300,000 images and 1.5 million Chinese descriptions. (Figure 7)

Figure 7 2017 AI Challenger dataset sample

4.2.2 Data Training Process

In this study, ResNet V1 50 is employed as the backbone model for extracting high-level semantic features from images. A critical modification must be made to the original ResNet V1 50 architecture: Remove the final fully connected (FC) layer of the network, or Replace the FC layer with an identity mapping (*i.e.*, skip the layer). The FC layer is designed to map features from the latent space to the sample space, serving as a feature fusion mechanism for classification tasks. However, since our objective is to obtain discriminative feature representations rather than classification probabilities, retaining the FC layer would introduce unnecessary dimensionality reduction and task-specific biases.

```
resnet50 = tv.models.resnet50(True).eval()    #用 resnet50 提取图像特征
del resnet50.fc
resnet50.fc = lambda x: x    #将全连接层替换为恒等映射
```

In the initial stage of training, the parameters of the ResNet V1 50 model remain fixed, serving solely as a static image encoding function. A trainable layer is appended to the end of the ResNet V1 50 model to transform image embedding vectors into the word embedding vector space. During this phase, the model optimizes the parameters for word embeddings, the trainable layers of ResNet V1 50, and the LSTM. In the second stage, all parameters – including those of ResNet V1 50 – are fine-tuned to optimize the entire encoding and decoding framework.

4.2.3 Text Generation Process

For a given trained model and image, the beam search strategy is employed to generate a natural language description of the image. The description is generated word by word, where at

each step t, it relies on the already generated sentence set of length t-1 to derive a new set of sentences of length t. In each step, only the top k candidates are retained, where k is the beam width, and experiments have shown optimal performance when $k = 3$.

4.2.4 Model testing

The trained model is evaluated using the *img2txt.py* script, which processes all images from Test Set A to generate corresponding Chinese captions. Figure 8 shows the example test results.

Figure 8 Image Captioning Model Test Result Samples

As illustrated in Figure 8, our trained model demonstrates the ability to identify the primary subjects within images, match them with corresponding actions, and synthesize these elements to generate textual descriptions of the images. Overall test results indicate that the model demonstrates a high accuracy rate in our study. However, as a Chinese language model, it occasionally exhibits issues such as inconsistent word order and redundant article usage. Since these image captions serve as input for the downstream text summarization module, the aforementioned limitations have minimal impact on the extraction of textual summaries. Based on this analysis, it is concluded that the model meets the required standards for deployment.

4.3 Word2Vec Model

This research proposes an integrative framework that synergizes the Word2Vec embedding architecture with the TextRank graph-based ranking algorithm for automated text summarization generation. Within this architecture, the Word2Vec model serves to encode discrete Chinese lexical units into distributed vector representations, capturing semantic relationships through neural language modeling. Concurrently, the TextRank algorithm implements a PageRank-inspired mechanism to computationally identify salient sentences within short-form video caption datasets, subsequently extracting these as core summary components through iterative voting processes.

4.3.1 Language Corpus Preprocessing

The Word2Vec model in this experiment was trained using the ChineseWiki corpus (Chinese Wikipedia Corpus) (Wang et al., 2012), with main text extracted through Wikipedia Extractor.

ChineseWiki is a collaboratively edited Chinese encyclopedia based on the wiki system, enabling broad public contributions. It contains over 990,000 Chinese entries covering most commonly used vocabulary. Wikipedia Extractor is a Python-implemented command-line tool designed to extract clean text from Wikipedia database dumps. It processes raw Wikipedia documents by removing wiki-specific syntax and formatting, outputting organized text files.

Our method was compared with existing title generation models. Experimental results demonstrate that our approach achieves significantly better performance than traditional rule-based methods in both coherence and informativeness.

```
PS git clone https://github.com/attardi/wikiextractor.git wikiextractor
PS wikiextractor/WikiExtractor.py   -b 1024M -o zhwiki_extracted   zhwiki-latest-pages-articles.xml.bz2
```

The above commands convert the ChineseWiki corpus into plain text. The -b 1024 MB option specifies a maximum file size of 1024 MB. In this experiment, the extracted plain text is approximately 1060 MB, so it is divided into two files: wiki_00 (1024 MB) and wiki_01 (36.7 MB). Initial experiments can be conducted on the smaller file before applying them to the larger one.

When using Wikipedia Extractor, certain preprocessing issues need to be considered that the tool removes many spaces and content within brackets, which may impact subsequent data analysis and model training. Wikipedia's Chinese dataset contains a mix of Simplified and Traditional Chinese, adding to the complexity of data processing. To ensure data accuracy and consistency, the extracted text must undergo thorough preprocessing before being used for model training.

(1) Traditional-to-Simplified Chinese Conversion with OpenCC

OpenCC (Open Chinese Convert) is an open-source project for Traditional-Simplified Chinese conversion, known for its accuracy and efficiency. OpenCC carefully distinguishes between one-to-many character mappings, ensuring accurate conversion. To convert text from Traditional to Simplified Chinese, use the following PowerShell commands:

```
PS opencc.py -i wiki_00 -o zh_wiki_00 -c zht2zhs.ini
PS opencc.py -i wiki_01 -o zh_wiki_01 -c zht2zhs.ini
```

(2) Text Cleaning with filter_wiki.py

The filter_wiki.py script cleans the text by replacing special symbols with quotation marks and removing empty brackets. Example code:

```
for line in file:
    p1 = re.compile('（）')                    line = p1.sub('', line)
    p2 = re.compile('《》')                    line = p2.sub('', line)
    p3 = re.compile('「')                      line = p3.sub('', line)
    p4 = re.compile('」')                      line = p4.sub('', line)
```

(3) Tokenization with Jieba

The *cut2words.py* script uses Jieba, an open-source Chinese segmentation tool, to tokenize the text. This step converts continuous sequences of Chinese characters into meaningful words, providing a foundation for further text processing and analysis.

4.3.2 Model Training

The Gensim (Khosrovian et al., 2008) library is used to train the Word2Vec model. Gensim is a powerful Python library designed for processing large-scale text data, extracting document topics, and measuring text similarity efficiently.

The *train_word2vec.py* script is used for training, with the following sample code:

```
sentences = word2vec.LineSentence(in_file)
model = word2vec.Word2Vec(sentences, size=200, min_count=5, workers=8)
model.save(out_file)
```

Three key parameters are used in training: 1) size: Determines the number of dimensions for word vectors (200 in this study); 2) min_count: Specifies the minimum frequency for a word to be included (set to 5 to filter out rare words); 3) workers: Controls parallel processing for training efficiency (set to 8 to speed up training). After training, the output includes: 1) The trained Word2Vec model; 2) Two word vector files.

```
[[-2.3125713   -1.6304954   -2.6447072   ... -0.7691004    0.21551     -0.6055662 ]
[-1.2669644   -0.36074203 -2.2054565   ... -0.687957     0.80676734   -2.0088034 ]
[-3.4282568   -1.6821556   -1.8597708   ... -1.187608     0.20196667   -1.7720966 ]
...
[ 0.04939042   0.02647006 -0.05875821 ...  0.01814336   0.06559797   -0.07578522]
[ 0.01686497 -0.03859647 -0.03823611 ... -0.03614297   0.09572794    0.00625494]
[ 0.01264876   0.01830564   0.00566043 ...  0.02357153 -0.0191332     0.00460972]]
```

4.3.3 Model Testing

The w2v_test.py script evaluates the model's performance, including: 1) Training time; 2) Model size; 3) Chinese word similarity tests. Example test code:

```
print(model.total_train_time)
print(model.wv)
print(model.wv.log_accuracy)
print(model.wv.similarity('因为','苹果'))
print(model.wv.similarity('篮球','足球'))
print(model.wv.similarity('中国','美国'))
print(model.wv.most_similar('狗'))
```

The results demonstrate that the model correctly calculates word similarity (values range from 0 to 1). The most_similar() function accurately finds related words. Test results:

```
956.8591299243604
<function KeyedVectors.log_accuracy at 0x000002A22ECCEA60>
0.1451199
0.6496723
0.4358243
[('猫', 0.7935165762901306), ('老鼠', 0.765432596206665), ('猴子', 0.7173890471458435),
('兔子', 0.7110925912857056), ('小狗', 0.6777244806289673), ('鸭子',0.6770710349082947),  ('青蛙', 0.6723482608795166),
('驴子', 0.6706522703170776), ('狐狸', 0.6651118397712708), ('猪', 0.6621549129486084)]
```

4.4 TextRank Algorithm Design

This experiment implements the TextRank algorithm based on the Word2Vec model, using two methods to compute sentence similarity.

4.4.1 Algorithm Modularization

In Table 2, the *Word2VecTextRank.py* script divides the TextRank process into several modules, with key functions.

Table 2 Word2vectTextRank.py Function Description

Function Name	Description
sentences_similarity(sents_1, sents_2)	Computes sentence similarity
cosine_similarity(vec1, vec2)	Computes cosine similarity between vectors
create_graph(word_sent)	Generates a similarity graph from sentence lists
compute_similarity_by_avg(sents_1, sents_2)	Computes average word vector similarity
calculate_score(weight_graph, scores, i)	Calculates sentence importance score
weight_sentences_rank()	Returns ranked sentence scores
diff(scores, old_scores)	Judge the changes
summarize(text, n)	Extracts top n most important sentences

By running *Word2VecTextRank.py*, the algorithm extracts the most important sentence as the short video title.

4.4.2 Algorithm Testing

The *Word2vectTextRank.py* script was executed using a web-captured 780-character introduction text of Beijing Institute of Technology as test input. The implementation first invokes summarize(text, 3) to extract the three most important sentences, followed by summarize(text, 1) to output the single highest-ranked sentence.

```
with open("text_1.txt", "r", encoding='utf-8') as txt_file:
        text = txt_file.read().replace('\n', '')
print(summarize(text, 3))
print(summarize(text, 1))
```

Figure 9 shows the test results:

```
Building prefix dict from the default dictionary ...
Loading model from cache C:\Users\13904\AppData\Local\Temp\jieba.cache
Loading model cost 1.012 seconds.
Prefix dict has been built successfully.
重要性最高的 3 个句子为:
['近年来，北京理工大学以高质量党建引领学校事业高质量发展，勇担强国建设新使命，持续加快推进“双一流”建设，精心培育一流人才，赤诚创造一流贡献，奋力建设一流大学，不断凝聚师生奋进力量，获评“全国文明校园”，办学事业实现跨越式发展，迈入高质量发展“新阶段”，稳居“双一流”高校第一方阵前列，进入历史最好发展时期。', '学校坚持瞄准国家重大战略需求和世界科技发展前沿锐意进取，曾创造第一枚二级固体高空探测火箭、第一台大型天象仪等新中国科技史上多个“第一”，始终在打造国家战略科技力量中展现担当作为，为科技创新、国家发展和社会文明进步作出了重要贡献。', '建校80余年来，北京理工大学坚持为党育人、为国育才，始终与党和国家同呼吸、共命运，走出了一条坚持德育为首、着力红专并举的“红色育人路”，一条立足国防特色、服务国家战略的“强军报国路”，一条勇于开拓创新、开放包容合作的“创新发展路”。']
重要性最高的 1 个句子为:
['近年来，北京理工大学以高质量党建引领学校事业高质量发展，勇担强国建设新使命，持续加快推进“双一流”建设，精心培育一流人才，赤诚创造一流贡献，奋力建设一流大学，不断凝聚师生奋进力量，获评“全国文明校园”，办学事业实现跨越式发展，迈入高质量发展“新阶段”，稳居“双一流”高校第一方阵前列，进入历史最好发展时期。']
```

Figure 9 TextRank Algorithm Test Results

Test result shows that the algorithm successfully identifies the most important sentence. However, the extracted sentence exceeds 50 characters, making it too long for a short video title. A post-processing step is needed to shorten the summary for better usability.

4.5 Text Summary Post-Processing

To address the issue of excessive text length in summaries, this study proposes a solution. In order to ensure that the generated short video titles are concise and aligned with the video content, key sentences describing the video content are first extracted. From these sentences, we select the most significant nouns, verbs, and place names based on their weight and combine them to form the final short video title.

4.5.1 Part-of-Speech Analysis

To implement post-processing of the text summary, we first need to perform word segmentation and part-of-speech analysis on the key sentences. This process is carried out using the HanLP natural language processing toolkit (Yan, 2022), specifically the interface (HanLP.parse(sentence, tasks='pos/863').pretty_print()). Here, "863" refers to a national standard established under the "863 Program" and adopted by the National Language Committee's corpus. The returned result is a new sentence that has undergone word segmentation and part-of-speech tagging.

A sample test result is as follows:

近年来/nt ， /w 北京/ns 理工/n 大学/n 以/p 高/a 质量/n 党建/j 引领/v 学校/n 事业/n 高/a 质量/n 发展/v ， /w 勇担/v 强国/n 建设/n 新/a 使命/n ， /w 持续/v 加快/v 推进/v “/w 双/a 一流/j ”/w 建设/n 。/w

The test results demonstrate that the 863 segmentation standard used in this study can accurately analyze the part-of-speech of each word in a sentence, which greatly contributes to summarizing short video titles.

4.5.2 Keyword Extraction

Next, keyword analysis is performed on the sentence, along with the calculation of the importance of each word. By using HanLP's keyword extraction function (HanLP.keyphrase_extraction (sentence)).

A sample test result is as follows:

{"'双一流"建设': 0.9384506940841675, '北京理工大学': 0.9113494157791138, '强国建设新使命':0.6882065534591675, "'双一流"': 0.6582499742507935, '高质量党建': 0.4676216244697571, '高质量': 0.41809314489364624, '强国建设':0.3577626347541809, '近年来': 0.3439931273460388, '一流': 0.23132982850074768, '建设':0.20067650079727173}

The test results show that a word's importance is related to its position and structure in the sentence. Words appearing earlier tend to have higher importance. Additionally, phrases generally have a higher weight than the individual words that compose them.

4.5.3 Title Summarization

Using the part-of-speech tagging results and the importance rankings, we can generate a summary of the key sentence to serve as the short video title. In this study, the noun with the highest importance in the key sentence is selected as the subject of the title, the most important verb is used as the predicate, and two other highly ranked nouns are used to describe the subject. The title follows the structure:

"Subject: Description 1 + Predicate + Description 2"

By running the *post_process.py* script, the conversion from text summaries of short video content to short video titles can be achieved.

A sample test result is as follows:

北京理工大学：高质量党建引领"双一流建设"

The test results indicate that the generated titles are within 20 characters, meeting the requirement for conciseness. Additionally, the keyword analysis method used in this study has a high accuracy rate, successfully identifying the main entity of the text summary and capturing its core content. In conclusion, the results obtained from this study align with the requirements for short video titles.

5 Conclusion

This paper implements the automatic generation of short video titles, detailing the entire process from theoretical research to experimental design. The main innovation of this work lies

in integrating part-of-speech analysis and keyword extraction techniques in natural language processing to convert lengthy video content summaries into concise video titles, making them easier for users to read. The primary research content of this paper includes:

(1) The research background of short video title generation, along with related academic studies and application achievements.

(2) An introduction to the deep learning frameworks and natural language processing toolkits used in the experimental process.

(3) The design and implementation of short video content summarization. This section elaborates on the neural networks and ranking algorithms employed in the experiment and their applications.

(4) The experiments and analysis of short video title generation. This section describes the entire experimental process from short video files to short video titles, including the tests conducted and analysis of the test results.

(5) Analyzing the generated short video titles and continuously refining the title generation method to ensure the titles meet user requirements.

6 Future Work

With the rapid development of the internet, the number of short videos is growing exponentially. According to statistics, the total number of users on short video platforms has exceeded one billion. The automatic generation of short video titles can help users save valuable time while filtering out non-compliant or inappropriate videos, demonstrating its social value.

Moreover, automatic short video title generation can assist individuals in rural areas by facilitating the promotion of local culture and develop regional tourism, thus driving economic growth. These facts indicate that automatic short video title generation has significant application potential and developmental prospects.

Although this study has achieved its expected goals, there are still areas for improvement. For instance, when generating video content descriptions, the recently popular "attention" model could be used to produce more accurate and reasonable descriptions. Additionally, the Chinese word segmentation tool employed in the final summarization process is not yet fully optimized, leading to the incorrect recognition of some common Chinese proper nouns. Furthermore, some generated titles exhibit disordered syntax, which does not align with natural reading habits. The authors hope that these issues will be further refined and resolved in future research.

Conflicts of interest

The author declares there is no conflict of interest.

References

Bradski, G., & Kaehler, A. (2000). OpenCV. Dr. Dobb's journal of software tools, 3(2).

Chang, L., Deng, X. M., & Zhou, M. Q. (2016). Convolutional neural networks in image understanding. Acta Automatica Sinica, 42(9), 13. https://doi.org/10.16383/j.aas.2016.c150800

Gao, M., Qi, D., Mu, H., & Chen, J. (2021). A transfer residual neural network based on ResNet-34 for detection of wood knot defects. Forests, 12(2), 212. https://doi.org/10.3390/f12020212

Goldberg, Y., & Levy, O. (2014). word2vec Explained: deriving Mikolov et al.'s negative-sampling word-embedding method. arxiv preprint arxiv:1402.3722. https://doi.org/10.48550/arXiv.1402.3722

Imambi, S., Prakash, K. B., & Kanagachidambaresan, G. R. (2021). PyTorch. Programming with TensorFlow: solution for edge computing applications, 87-104.

Jiang, F., & Zhang, Y. J. (2003). Scene segmentation and indexing and summarization of news videos. Chinese Journal of Computers, 26(7), 859–865. https://doi.org/10.3321/j.issn:0254-4164.2003.07.013

Khosrovian, K., Pfahl, D., & Garousi, V. (n.d.). (2008). GENSIM 2.0: A Customizable Process Simulation Model for Software Process Evaluation. Making Globally Distributed Software Development a Success Story, 294–306. https://doi.org/10.1007/978-3-540-79588-9_26

Ladias, A., Karvounidis, T., & Ladias, D. (2021). Classification of the programming styles in scratch using the SOLO taxonomy. Advances in Mobile Learning Educational Research, 1(2), 114-123. https://doi.org/10.25082/AMLER.2021.02.006

Langville, A., & Meyer, C. (2004). Deeper Inside PageRank. Internet Mathematics, 1(3), 335–380. https://doi.org/10.1080/15427951.2004.10129091

Liu, X., Zhu, Z., Fu, T., Chen, J., & Jiang, Y. (2021). Corpus annotation system based on HanLP Chinese word segmentation. In The 2nd International conference on computing and data science (pp. 1-17).

Mihalcea, R., & Tarau, P. (2004). Textrank: Bringing order into text. In Proceedings of the 2004 conference on empirical methods in natural language processing (pp. 404-411).

Neil, D., Pfeiffer, M., & Liu, S. C. (2016). Phased lstm: Accelerating recurrent network training for long or event-based sequences. Advances in neural information processing systems, 29. https://doi.org/10.48550/arXiv.1610.09513

Shu, N., Liu, B., Lin, W., & Li, P. (2019). Survey of Distributed Machine Learning Platforms and Algorithms. Computer Science, 46(3), 9-18. https://doi.org/10.11896/j.issn.1002-137X.2019.03.002

Suarez, O. D., Carrobles, M. D. M. F., Enano, N. V., García, G. B., Gracia, I. S., Incertis, J. A. P., & Tercero, J. S. (2014). OpenCV Essentials. Packt Publishing Ltd.

Szegedy, C., Ioffe, S., Vanhoucke, V., & Alemi, A. (2017). Inception-v4, Inception-ResNet and the Impact of Residual Connections on Learning. Proceedings of the AAAI Conference on Artificial Intelligence, 31(1). https://doi.org/10.1609/aaai.v31i1.11231

Uygun, D. (2024). Teachers' perspectives on artificial intelligence in education. Advances in Mobile Learning Educational Research, 4(1), 931-939. https://doi.org/10.25082/AMLER.2024.01.005

Venugopalan, S., Rohrbach, M., Donahue, J., Mooney, R., Darrell, T., & Saenko, K. (2015). Sequence to Sequence – Video to Text. 2015 IEEE International Conference on Computer Vision (ICCV). https://doi.org/10.1109/iccv.2015.515

Wang, Z., Wang, Z., Li, J., & Pan, J. Z. (2012). Building a Large Scale Knowledge Base from Chinese Wiki Encyclopedia. The Semantic Web, 80–95. https://doi.org/10.1007/978-3-642-29923-0_6

Wilson, M. A., Zhou, Z., & Frank, R. (2023). Subject-verb agreement with Seq2Seq transformers: Bigger is better, but still not best. Proceedings of the Society for Computation in Linguistics, 6(1), 278-288.

Wu, J., Zheng, H., Zhao, B., Li, Y., Yan, B., Liang, R., Wang, W., Zhou, S., Lin, G., Fu, Y., Wang, Y., & Wang, Y. (2019). Large-Scale Datasets for Going Deeper in Image Understanding. 2019 IEEE International Conference on Multimedia and Expo (ICME). https://doi.org/10.1109/icme.2019.00256

Xiong, Y. (2014). Moving object extraction based on background subtraction and inter-frame difference method. Computer Era, 3, 4. https://doi.org/10.3969/j.issn.1006-8228.2014.03.013

Yan, B. (2022). Graph construction based on HanLP keyword extraction and syntactic analysis. Electronic Component and Information Technology, 9, 77–80.

Zhang, C., & Woodland, P. C. (2015). Parameterised sigmoid and reLU hidden activation functions for DNN acoustic modelling. Interspeech 2015. https://doi.org/10.21437/interspeech.2015-649

Zhang, Z. X., Wang, H., & Xu, D. (2014). The rise and trends of "mobile short video social applications". China Journalist, 2, 107–109. https://doi.org/10.3969/j.issn.1003-1146.2014.02.054