Regular paper
Journal of the KIECS. pp. 81-88, vol. 16, no. 1, Feb. 28. 2021, t. 105, pISSN 1975-8170 | eISSN 2288-2189
http://dx.doi.org/10.13067/JKIECS.2021.16.1.81

음성인식을 이용한 자막 자동생성 시스템

손원섭* · 김응곤**

Subtitle Automatic Generation System using Speech to Text

Won-Seob Son* · Eung-Kon Kim**

요 약

최근 COVID-19로 인한 온라인 강의 영상과 같은 많은 영상이 생성되고 있는데 노동 시간의 한계와 비용의 부족 등으로 인해 자막을 보유한 영상이 일부분에 불과하여 청각장애인들의 정보 취득에 방해 요소로 대두되고 있다. 본 논문에서는 음성인식을 이용하여 자막을 자동으로 생성하고 종결 어미와 시간을 이용해 문장을 분리하여 자막을 생성함으로써 자막 생성에 드는 시간과 노동력을 줄일 수 있도록 하는 시스템을 개발하고자 한다.

ABSTRACT

Recently, many videos such as online lecture videos caused by COVID-19 have been generated. However, due to the limitation of working hours and lack of cost, they are only a part of the videos with subtitles. It is emerging as an obstructive factor in the acquisition of information by deaf. In this paper, we try to develop a system that automatically generates subtitles using voice recognition and generates subtitles by separating sentences using the ending and time to reduce the time and labor required for subtitle generation.

키워드

STT, Scheduling, Automatic subtitle generation, STT post-processing, Sentence analysis
STT, 스케줄링, 자막 자동 생성, STT 후처리, 문장 분석

Ⅰ. 서 론

최근 미디어 시장의 성장 및 COVID-19로 인한 온라인 강의 영상 등 영상의 소비가 많아지고, 그에 따라 소규모 영상 제작자나 1인 영상 제작자들은 시간을 들여 자막을 생성하고 있지만 짧은 시간의 급격한 변화와 1인 영상 제작의 노동 시간의 한계 등으로 인해 자막이 없는 영상들이 많이 생성되고 있다[1-2].

자막이 없는 영상은 자막이 있는 영상에 대비해 접근성이 떨어져서 청각장애인들의 정보 취득 방해 요소로 작용하고 있으므로 자막을 빠르게 생성해 줄 수 있는 시스템이 필요한 실정이다[3-6].

* 순천대학교 컴퓨터공학과(thsdnjstjq1@naver.com)
** 교신저자 :순천대학교 컴퓨터공학과
• 접 수 일 : 2020. 12. 04
• 수정완료일 : 2021. 01. 10
• 게재확정일 : 2021. 02. 17
• Received : Dec. 04. 2020, Revised : Jan. 10, 2021, Accepted : Feb. 17, 2021
• Corresponding Author : Eung-Kon Kim
Dept. of Computer Engineering, Sunchon National University.
Email : kek@scnu.ac.kr

II.자막 자동생성 시스템 설계

2.1 자막 자동생성 시스템 구성

자막 자동생성 시스템의 구성은 그림 1과 같다. 클라이언트, 서버 구성으로 이루어져 있고 다수의 클라이언트가 한 개의 서버로 요청을 보내는 구조로 이루어져 있다. Speech to Text(: STT) 요청처리 API와 STT 결과 처리 API로 2개의 API와 STT 처리기로 구성되어 있다.

STT 요청처리 API는 STT 작업 요청이 들어오면 STT 작업 목록을 생성하여 STT 처리기에 작업 목록을 넣고 클라이언트에 작업 식별키 목록을 반환한다. STT 결과 처리기는 클라이언트로부터 작업 식별키를 받아. STT 처리기에 해당 식별키로 작업을 찾고 그에 따른 작업 결과를 반환한다.

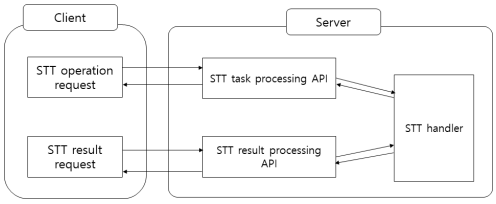

그림 1. 자동 자막 생성시스템 구성도
Fig. 1 Diagram of automatic subtitle generation system

2.2 STT 작업 요청처리

STT 작업 요청처리 순서도는 그림 2와 같다. 클라이언트로부터 수신된 영상을 1분 단위로 나누고 분할된 각각의 영상에 고유한 작업 식별키를 부여한다. 부여한 작업 식별키를 모아 작업 목록을 생성하여 클라이언트로 작성하고, 작업 처리기에 작업을 등록한다[7].

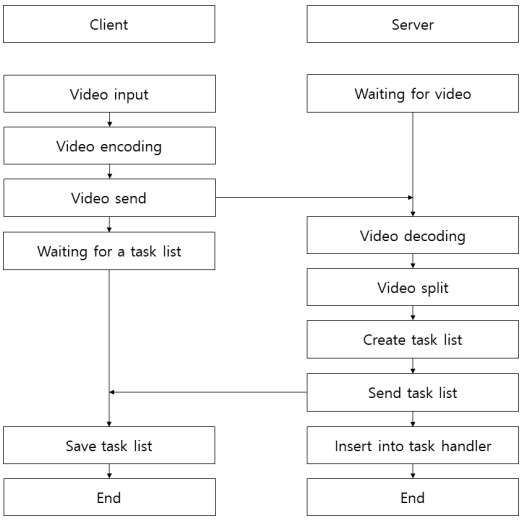

그림 2. STT 작업 요청처리 순서도
Fig. 2 STT task request processing flowchart

2.3 STT 처리기

STT 처리기는 STT 작업 처리부와 STT 작업 스케줄링 부분으로 나누어진다. STT 작업 처리부는 그림 3과 같은 구성으로 이루어져 있다.

STT 작업 처리부는 그림 3과 같은 순서로 구성되어 있다. STT 작업이 입력되면 semaphore 카운트가 0이 아닐 때까지 대기 하다가 0이 아니게 되면 카운트를 감소하고 작업을 진행한다. 작업이 종료되면 다시 semaphore 카운트를 증가시킨다.

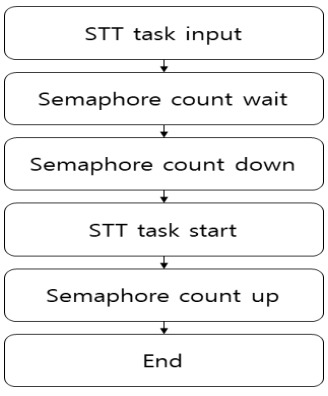

그림 3. STT 작업 요청처리 순서도
Fig. 3 STT task request processing flowchart

STT 서버가 많은 양의 STT 작업을 동시에 처리해야 하는 경우, 서버의 스레드 수보다 STT 작업량이 많으면 동시에 처리하기 위해 문장 재구성의 자주 일어나게 되는데, 이때 발생하는 오버헤드가 상당히 커서 비효율적으로 작업이 진행된다[8]. 따라서, 서버의 스레드 수보다 작은 수의 세마포어를 카운트를 설정하여 무장 재구성이 최소로 일어나게 하여 오버헤드를 줄일 필요가 있다.

Semaphore만을 사용하여 작업을 관리하는 경우 동시에 진행될 수 있는 작업에 제한이 생겨 대기하는 작업이 생기게 되고 대기하는 작업끼리 semaphore 카운트를 가지기 위해 경쟁하게 된다. 또한, 지속해서 semaphore 카운트를 가지지 못해 기아 상태에 빠지게 되는 작업이 생길 위험이 있다. 따라서, 대기 중인 작업을 관리해줄 필요가 있다.

대기 중인 작업을 처리하기 위한 STT 작업 스케줄링의 순서도는 그림 4와 같다. 스케줄링을 위해 작업 대기 큐와 대기 작업 목록을 구성되어 있다. 작업 대기 큐에 남아 있는 작업이 없으면 대기 작업 목록을 우선순위에 따라 정렬을 하여 작업 대기 큐에 삽입한다. 작업 대기 큐는 STT가 완료되는 시점에 STT 처리기로 작업을 반환한다.

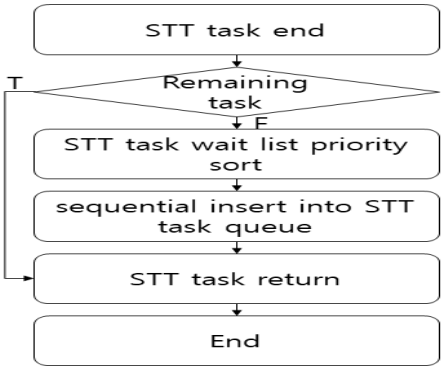

그림 4. STT 작업 스케줄링 순서도
Fig. 4 Flowchart of STT task scheduling

2.4 STT 결과 처리

STT 결과 처리의 순서도는 그림 5와 같다. STT 작업 식별 키와 함께 결과 요청이 들어오면 작업 식별 키로 작업 결과를 조회하여, 작업 진행 상황 상태와 그 결과를 반환한다.

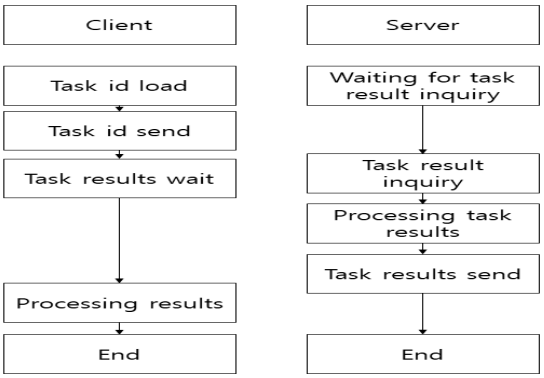

그림 5. STT 결과 처리의 순서도
Fig. 5 Flowchart of STT result processing

2.5 STT 후처리

STT 작업이 완료되면 시작 시각, 종료 시각, 단어로 이루어져 있는 단어들로 반환되기 때문에 후처리 없이 자막으로 바로 쓰기는 어려우므로 후처리가 필요하다.

STT 후처리는 그림 6과 같이 구성된다. 작성 중인 문장이 없으면 새로운 문장을 생성한 후 그 문장에 단어를 저장한다. 작성 중인 문장이 있으면 추가할 단어가 문장의 시작을 의미하는 단어이거나 직전에 추가된 단어와의 시간 간격을 비교하여 지정된 시간 간격 이상이면 새로운 문장을 생성하고 그 문장에 저장한다. 시작을 의미하는 단어가 아닌 경우 현재 문장에 단어를 저장하고 저장된 단어가 문장의 끝을 의미할 때 새로운 문장을 작성하여 후에 들어오는 단어들은 새로운 문장에서 저장되도록 설계하였다[9].

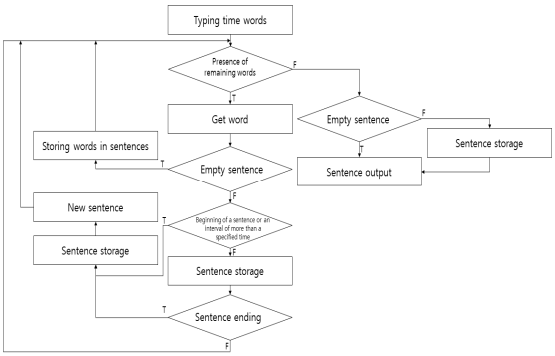

그림 6. STT 후처리 순서도
Fig. 6 STT post-processing flowchart

III. 자막 자동생성 시스템 구현

3.1 자막 자동 생성시스템 구현환경

자막 자동 생성시스템 구현환경은 그림 8과 같다. Docker를 이용해서 어느 환경이든 상관없이 같은 서버 환경에서 실행할 수 있고, Nginx를 이용한 역방향 프록시를 구축하여 Kestrel이 처리하기 힘든 도메인 네임 필터링과 Docker Network 설정 방식에 따라 Client의 IP가 수집이 어려운 경우에도 Proxy header를 통해 수집할 수 있게 설계하였다[10-12].

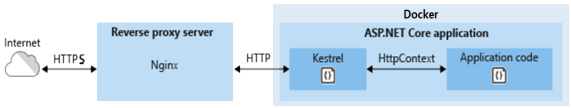

그림 7. 자막 자동 생성시스템 구현환경
Fig. 7 Implementation environment of subtitle automatic generation system

3.2 STT 요청 결과

STT 요청의 결과는 그림 8과 같다. 입력된 음성파일을 FFmepge를 이용해 wav를 파일로 변환하고, 이 파일을 1분 단위로 변환하여 STT 서버로 전송하고 각각의 TaskID를 반환한다.


```
{
    "requsetId": "463113d5-93c1-4a55-9db1-08d8734cb125",
    "order": 33,
    "taskIds": [
        "0-463113d5-93c1-4a55-9db1-08d8734cb125",
        "1-463113d5-93c1-4a55-9db1-08d8734cb125",
        "2-463113d5-93c1-4a55-9db1-08d8734cb125",
        "3-463113d5-93c1-4a55-9db1-08d8734cb125",
        "4-463113d5-93c1-4a55-9db1-08d8734cb125",
        "5-463113d5-93c1-4a55-9db1-08d8734cb125",
        "6-463113d5-93c1-4a55-9db1-08d8734cb125",
        "7-463113d5-93c1-4a55-9db1-08d8734cb125",
        "8-463113d5-93c1-4a55-9db1-08d8734cb125",
        "9-463113d5-93c1-4a55-9db1-08d8734cb125",
        "10-463113d5-93c1-4a55-9db1-08d8734cb125",
        "11-463113d5-93c1-4a55-9db1-08d8734cb125",
        "12-463113d5-93c1-4a55-9db1-08d8734cb125",
        "13-463113d5-93c1-4a55-9db1-08d8734cb125",
        "14-463113d5-93c1-4a55-9db1-08d8734cb125",
        "15-463113d5-93c1-4a55-9db1-08d8734cb125",
        "16-463113d5-93c1-4a55-9db1-08d8734cb125",
        "17-463113d5-93c1-4a55-9db1-08d8734cb125",
        "18-463113d5-93c1-4a55-9db1-08d8734cb125",
        "19-463113d5-93c1-4a55-9db1-08d8734cb125",
        "20-463113d5-93c1-4a55-9db1-08d8734cb125",
        "21-463113d5-93c1-4a55-9db1-08d8734cb125",
        "22-463113d5-93c1-4a55-9db1-08d8734cb125",
        "23-463113d5-93c1-4a55-9db1-08d8734cb125",
        "24-463113d5-93c1-4a55-9db1-08d8734cb125",
        "25-463113d5-93c1-4a55-9db1-08d8734cb125",
        "26-463113d5-93c1-4a55-9db1-08d8734cb125",
        "27-463113d5-93c1-4a55-9db1-08d8734cb125",
        "28-463113d5-93c1-4a55-9db1-08d8734cb125",
```


그림 8. STT 요청 결과
Fig. 8 STT request result

3.3 STT 결과

STT 분석 결과 요청의 결과는 그림 9와 같다. STT 분석이 완료되면 Status와 result를 반환하고, 한 문장을 이루는 단위를 배열로 묶어서 반환한다.


```
{
    "status": "Done",
    "result": [
        [
            {
                "confidence": 0.484,
                "text": "아싸",
                "start": 121.59,
                "end": 122.153
            },
            {
                "confidence": 1,
                "text": "드디어",
                "start": 122.25,
                "end": 122.7
            }
        ],
        [
            {
                "confidence": 0.995,
                "text": "너한테만",
                "start": 126.24,
                "end": 126.81
            },
            {
```


그림 9. STT 결과
Fig. 9 STT result

3.3 STT 스케줄링

STT 분석 스케줄링 과정은 그림 10과 같다. RequsetTIme이 NULL이 아닌 레코드는 STT 분석이 실행 중인 레코드이고, ReponseTime이 NULL이 아닌 레코드는 작업이 완료되어 분석된 Text를 가지고 있는 레코드이다. 마지막으로 RequsetTime, ResponsTime 둘 다 NULL인 레코드는 작업 대기 중인 항목이다.

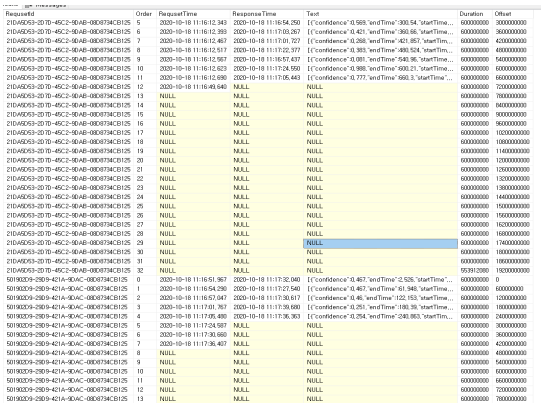

그림 10. STT 스케줄링 결과
Fig. 10 STT scheduling result

3.4 문장 분석 구현

문장 분석 구현은 그림 11과 같다. 가장 첫 번째 단어로 문장을 시작하고, 그 후 공백이나 개행 등 빈 문자열만 있는 단어일 경우 무시하고 시작 어두 (‘야’ 혹은 ‘왜’)인 경우에 새로운 문장을 시작한다. 시작 어두가 아니라면 현재 문장에 단어를 추가하고 종결 어미(‘까’, ‘다’, ‘요’ 등)일 때, 현재 문장을 종료한다. 이 과정을 입력받은 모든 단어에 적용한다.

```
public static List<List<T>> Parser<T>(List<T> words) where T : Word
{
  var output = new List<List<T>>();

  if (words.Count == 0)
  {
    return output;
  }

  var subtitles = new List<T>
  {
    words[0]
  };

  for (int i = 1; i < words.Count; i++)
  {
    var item = words[i];
    if (string.IsNullOrWhiteSpace(item.Text)) continue;

    if (subtitles.Count == 0)
    {
      subtitles.Add(item);
      continue;
    }

    if (StartCheck(item) || item.Start - words[i - 1].End > 1.0)
    {
      output.Add(subtitles);
      subtitles = new List<T>();
    }

    subtitles.Add(item);

    if (EndCheck(subtitles))
    {
      output.Add(subtitles);
      subtitles = new List<T>();
    }
  }

  if (subtitles.Count > 0)
  {
    output.Add(subtitles);
  }

  return output;
}
```

그림 11. 문장 분석 구현
Fig. 11 Implement sentence analysis

Ⅳ. 테스트

본 논문에서 제안한 자막 자동생성 시스템의 평가 항목은 표1과 같다.

표 1. 평가항목
Table 1. Evaluation items

Evaluation item	target	Result
1. first response time	≤30sec	11.41sec
2. last response time	≤600sec	457.51sec
3. average response time	≤60sec	34.44sec

테스트 결과로 STT 첫 번째 응답 시간이 목표치 30 초 이하인 11.41 초, STT 마지막 응답 시간의 목표치 600 초 이하인 457.51 초, 평균 응답 시간의 목표치인 60 초 이하인 34.44 초임을 확인하였다.

단순 STT와 문장 분석 STT의 비교는 표 2와 같다. Google의 STT 엔진은 문장의 종결에 마침표, 문단의 끝에 문단 나눔을 두어 분리하는 반면, 문장 분석 STT는 자동 자막 생성 STT는 종결 어미와 시간 간격까지 포함하여 문장을 분리하였다. 결과로 종결 어미에 의한 문장 분리가 구글은 8개, 자막 자동생성 시스템은 8개로 같았고, 시간에 의한 분리는 각각 0개, 2개로 문장을 더욱 세밀하게 나눌 수 있어서 자막 생성을 보다 효율적으로 할 수 있었다.

표 2. STT 결과 비교
Table 2. STT result comparison

STT	Sentence separation	Time	Total Split
Google	8	0	8
Subtitle automatic generation	8	2	10

Ⅴ. 결 론

본 논문에서는 COVID-19와 미디어 시장의 성장에 따라 쉽고 빠르게 자막 생성할 수 있는 기술의 필요

성 증가에 따라, 사용자가 데이터를 전처리, 후처리할 필요 없이 사용 가능한 음성인식을 이용한 자막 자동생성 시스템에 대하여 제안하였다.

본 논문에서 제안한 자막 자동생성 시스템은 많은 사용자가 한 개의 서버를 이용해 생성된 자막을 받아보는 구조로써, 영상을 한 번에 번역해서 결과를 돌려주는 것이 아니라 영상을 앞에서부터 차례대로 분석하여 돌려줌으로써, 사용자의 빠른 반응성을 기대할 수 있다. 또한, 인코딩 과정을 서버에서 수행함으로써 사용자들은 영상만 올리면 되기 때문에 쉽게 사용할 수 있게 구현하였다.

본 논문에서 제안한 시스템은 기존에 있는 STT와 비교하면 자막 생성에 있어 불필요한 데이터를 최소로 전송하고, 문맥을 구분한 결과를 돌려주기 때문에 사용자들의 자막 생성에 편리하다.

향후 연구과제로, 현 시스템은 영상의 길이로 나누어 STT를 수행하지만. 음성이 있는 구간만 검출하여 STT를 수행하면 많은 시스템 리소스를 아낄 수 있다고 추측되어, STT를 수행하기 전에 음성 구간 검출 수행이 필요하다.

또한, 문장 분석을 휴리스틱 기법을 이용하여 분석 중인데, 이는 각각의 사용자 고유의 문장 구분법을 만족하게 할 수는 없으므로, 개인별 모델을 구현하여 사용자 맞춤형 문맥 구분법을 적용이 필요하다.

References

[1] Y. Baek, H. Lee, and J. Oh, "A Study on the Near Field IoT Medical Receipt System Based on Uncontact," *J. of the Korea Institute of Electronic Communication Sciences*, vol. 18, no. 1, 2020, pp. 73-110.

[2] Y. Sun, "A Semantic Diagnosis and Tracking System to Prevent the Spread of COVID-19," *J. of the Korea Institute of Electronic Communication Sciences,* vol. 15, no. 3, 2020, pp. 611-616.

[3] J. Park, "How do Creators Work? A Critical Study on the Production Experience of Personal Media," *Journal of media economics & culture,* vol. 18, no. 1, 2020, pp. 73-110.

[4] S. Kim, "Machine Learning based Automatic Caption Generation System for Speaker Diarization," Docate, *Graduate School of Korea University of Technology and Education,* 2019.

[5] J. Choi, "Independent component analysis based on frequency domain model for speech source signal extraction," *J. of the Korea Institute of Electronic Communication Sciences,* vol. 15, no. 5, 2020, pp. 807-812.

[6] Y. Kim and M. Chung, "Improving Performance of Continuous Speech Recognition Using Error Pattern Training and Post Processing Module," *Journal of Korean Information Science Society,* vol. 27, no. 1B, 2000, pp. 441-443.

[7] K. Ok, J. Park, W. Lee, and J. Ho, "An Improved Adaptive Job Allocation Method for Multiprocessor Systems," *Journal of The KIPS Transactionsty,* vol. 6, no. 6, 1999, pp. 1502-1510.

[8] Y. Son, "An Extended Conflict-Resolution Method using Multiple Semaphor Scheme," *Journal of Bulletin of the Institute for industrial Science,* vol. 15, no. 2, 1992, pp. 139-147.

[9] E. Choi, "A Study of Shortened Conclusive-Endings for Korean Language Education: Focused on Compound Forms of Double Conclusive-Endings," *Journal of the International Network for Korean Language and Culture,* vol 8, no 1, 2011, pp. 205-230.

[10] J. Kim and S. Kim, "Server construction for game engine development using Node.js+Nginx," *Journal of the Korean Society of Information Technology,* vol. 13, no. 122, 2015, pp. 109-114.

[11] Y. Bea, S. Jung, and W. Soh, "Comparative Analysis of the Virtual Machine and Containers Methods through the Web Server Configuration," *Journal of the Korea Institute of Information and Communication Engineering,* vol 18. no. 11, 2014, pp. 2670-2677.

[12] S. Shin, K. Kim, J. Jang, W. Sohn, and C. Park, "Securing Reverse Proxy Server for defending DDOS attack," *Korean Society of Electronics Engineers Conference,* Pyeongchang, Korea, June, 2003, vol. 2003, no. 11, pp. 430-433.

저자 소개

손원섭(Won-Seob Son)

2015년 3월 ~ 현재 순천대학교 컴퓨터공학과 재학

※ 관심분야 : 음성인식, 영상처리, 컴퓨터 그래픽스

김응곤 (Eung-Kon Kim)

1980년 2월 : 조선대학교 공학사
1986년 2월 : 한양대학교 공학석사
1992년 2월 : 조선대학교 공학박사
1993년 3월 ~ 현재 : 순천대학교 컴퓨터공학과 교수

※ 관심분야 : 영상처리, 컴퓨터 그래픽스, 멀티미디어, HCI