Screencast search
Functional Specification

Michael Wall 13522003
Jordan Healy 13379226

Monday 5th December, 2016

Contents

1 Introduction

The idea for this search system is to enable users to search presentation materials, specifically videos, with greater ease. At present, if a user wants to look for information on a topic, they may come across a video presentation on the topic. The video may be too long to watch all of if the user is only looking for a specific piece of information. The system aims to allow a user to search theses styles of video to get straight to the section which is relevant to their information need.

The need for this system comes from researching college material using external sources. Sometimes these come in the form of lectures and screencasts from other colleges. These lectures may have the information that you are looking for but are too long to regard as a usable resource.

These lectures are usually in video format with presentation slides as the video and audio from the lecturer overlaid on the video. The system will allow the user to search the contents of the lecture slides used by the lecturer and also to search the voiceover presented by the lecturer for extra information that may not have been included in the slides.

The plan for this system is to use image processing to recognise text boxes or lecture slides which contain strings of text from keyframes in the video. The system will then use Optical Character Recognition (OCR) to convert the images into searchable text. In addition, the audio will be parsed using Automatic Speech Recognition (ASR) to retrieve searchable text. The users search request will then be applied to the output of the above functions to find sections in the video which may be relevant. These sections will be highlighted to the user so that they can jump to these points in the video.

2 Functional Description

Our system will use a combination of Optical Character Recognition and Automatic Speech Recognition to process a selected video. These are detailed further below.

3 Optical Character Recognition (OCR)

For OCR we must split the video up into a series of images. These images will be frames of the video taken at regular intervals. If two or more images from these

frames are a said to be 90% similar then we assume we have a duplicate slide from the presentation video. For this, we remove these duplicates.

3.1 OCR - Preprocessing

Text localisation will be used to identify the position of potential text regions from an image.

3.1.1 Canny filters

Purpose Identify text blocks, where the blocks are defined as rectangular regions containing one or more lines of text [1].

Advantages Using probability for finding error rate.
Localization and response.
Improving signal to noise ratio.
Better detection especially in noise conditions. [3]

Disadvantages Complex Computations.
False zero crossing problem.
Time consuming [3].

3.1.2 Baseline extraction

Purpose Isolate and extract these potential text regions.

Advantages Easy to use.
It works well with the printed text [4].

Disadvantages Difficulty with handwriting text especially with handwritten words [4].

Text verification will be used to classify text lines into actual text regions.

3.1.3 Normalisation

Purpose Create all text regions equal in size [1].

Advantages Reduces the number of false alarms and creates a more precise location for each text region [1].

Disadvantages If normalisation processes is not precise, output could be worse than input.

3.1.4 Align images, remove noise and binarize them

Purpose So that the most effective OCR can be carried out [1].

Advantages Will generate better results for OCR.

Disadvantages If removing too much noise may remove vital pixels needed for OCR.

3.2 OCR - Algorithms

OCR algorithms fall into two categories, matrix matching or feature extraction (aka Intelligent character recognition).

Matrix matching works by comparing what the OCR sees as a character with a library of character matrices. When an image matches one of these prescribed matrices of dots, the computer labels that image as the corresponding ASCII value.

Feature extraction does not use strict matching to its prescribed templates (like matrix matching does). The computer looks for general features like open areas, closed shapes, diagonal lines and line intersections. This process is used more for written text and requires machine learning to implement.

3.2.1 Matrix matching

Advantages Easier and more simple to implement.

Disadvantages Not great against different styles of text.
Not great with noise [2].
Not great with rotated characters.

3.2.2 Feature extraction

Advantages More versatile.

Disadvantages Harder to implement.

We decided upon matrix matching for our OCR algorithm. This is because feature extraction is used mainly for handwritten text, which doesnt really apply to our use case since we can assume presentation slides will be in the form of typed text. We also feel that we can overcome the disadvantages of matrix matching within pre-processing.

BEGINNOTE: possible removal of detail above and below

3.3 Text localisation

3.3.1 Canny filters

A text block (identified by Canny filters) can be classified by being either a horizontal or vertical block. An example of a text block looks like so.

When the filters are applied to the whole image, we create two dilation images, horizontal dilation and vertical dilation. To create a dilation image for the text from the original image, we simply AND them together.

The entire procedure till this step is fast and extracts text regions irrespective of its background. Recall is high but there maybe errors due to sharp edges, slanting striped or in small areas of background. Next, we must extract these potential text regions and normalise them, using baseline extraction, so that it is suitable for OCR. The process of baseline extraction starts with calculating the ANDd images Y-axis projection, h(y). That is the number of text pixels in line y on the image. We then use three splitting algorithms to extract the regions of the image that have text.

- Variable Length Splitting: Remove text whose length is usually shorter than that of typical text strings. E.g. remove text that are abbreviations, like a sign reading C.V.S. We select the Y-coordinate y0 with maximum absolute derivative for any each text region. If this value is above a given threshold, tg, and h(y0) is below 50% of the length of the longest line in the region, then the region is split at line y0.
- Equal Length Splitting: When a region consists of two text lines of similar lengths, it may be split using Otsus thresholding method [2]. Otsus method finds the threshold (line y0) that minimizes the intra-class variance of the two text lines. Then, like the method before this, if h(y0) is less than 50% the longest line in this region, we split the region at line y0.
- Baseline Refinement: If a region cannot be split by the previous two algorithms, it is assumed that it may contain only one text line. We refine the

top and bottom boundaries (baselines) of the region to obtain a more precise location.

3.3.2 Text Verification

Each text line is normailised by height and its features are classified into four categories:

- Grayscale Spatial Derivative Features: All features are extracted from 16x16 windows sliding over the image grid. This feature measures the contribution of contrast for a given window.
- Distance Map Features: Distance Map is a feature image that only relies on the position of strong edges on the image. Since grayscale values of both characters and background may vary, contrast is background dependent.
- Constant Gradient Variance Features: Since distance maps are based on edges, they involve thresholds that need to be tuned. To avoid using any thresholds, constant gradient variance features are used.
- DCT Coefficients Features: DCT coefficients are computed over 16 16 blocks using fast DCT algorithms. This frequency domain feature is widely used in JPEG and MPEG compression techniques.

One final step before passing the resulting images to OCR is to align all text fields, remove background noise and binarize them.

Now that our image has been pre-processed we can pass it to our OCR. Our matrix matching algorithm will work as follows and is an expansion on the procedure described by Junaid Tariq, Umar Nauman and Muhammad Umair Naru [4]:

- Save a database of all valid characters. Each entry will consist of the character name, height (the difference between the first and last black pixel encounter from top to bottom), width (the difference between the leftmost and rightmost black pixel), checksum (total number of pixels of that character) and font style. E.g. B, 14, 8, 71, Ariel
- We extract these details for each char from the binarized image, and query the database.
  - First, we use hard matching to find the the exact values in the database

– If a match is not found, we query again but using soft matching. A value 2 for each criteria

To address the issues of typical matrix matching, we will add the same characters different font styles to our database. This should help the OCR in finding different text styles.

ENDNOTE: Possible reduction in detail.

4 Automatic Speech Recognition (ASR)

[1]

With current technology in automatic speech recognition it is not possible to generate a perfect transcription of audio. This is due to a combination of issues related to acoustics, language models and phonetics. The generated index data can be searched through, however it will have errors which will cause some search terms not to be found, even in a relevant piece of audio dialogue. This will in turn reduce the Mean Average Precision of the overall system. However, this might be tolerable if not all of the data is required to be relevant in a search, as long as we can have some keywords that are important identified. For example, if the lecturer speaks about a topic and mumbles through some of the sentences, but mentions one of our keywords. If this is recognised then it may be irrelevant whether or not the rest of the sentence was heard correctly. In our case, the direct translation of the audio is not necessarily what we are trying to obtain either, just a pointer to a section that might be relevant. The user can then navigate to this section and determine if it is of relevance for themselves.

For our videos, we will be processing them directly through YouTube, as this is where the majority of our target data is located. YouTube provides a developer application programming interface to access a caption track[2] on a given video. This API allows the captions to be retrieved using a HTTP GET request.

In the best case scenario, the video will have been transcribed by the video uploader, or contributed by a viewer. This will provide a very reliable audio transcription to search through as phonetic stumbles such as "um", "ah" and so on may be removed through stop word removal.

In cases where captions have not been manually provided, captions are generated using Google's speech-to-text engine. This engine works for over 80 languages[3]. In my experience, this API provides a reasonably reliable level of accuracy, even in noisy or imperfect audio environments.

4.0.1 Algorithms

We will process the audio data as follows:

- The search request will be broken into terms using porter stemming algorithm.
- The caption data will also be passed over using porter stemming to get all of the terms.
- Appearances of the search terms will be marked in the caption data.
- The marked sections will be compressed in a 0 - 100 range in the form of a heat map.
- This heatmap will allow the user to see where their search terms occur most frequently in the caption data, and they can use this to navigate to relevant sections in the video.

4.0.2 Pros and cons

Trusting Google's Machine Learning has it's advantages and disadvantages. For one, they are using Machine Learning to constantly improve their accuracy in translation, and they also have access to possibly the largest set of training data for such a task. This is also a built in solution to the videos, and does not require extra computational resources for the user to process a video. This preprocessing is done by YouTube when a video is uploaded.

On the other hand, the accuracy of this method is not as accurate as if we were to have videos manually transcribed. This would vastly improve our search capability and accuracy, but at the expense of costly man hours. It would also limit the number of videos our search application could be used for.

4.0.3 Assumptions

We are making assumptions about the data which a user want's to search. Such assumptions are that the videos the user wants exist on YouTube and not on a college website. It also assumes that the API provided by YouTube will be available indefinitely in the future.

5 Implementation

Before we can use the OCR and the ASR there is pre-processing to be done. The entire system process will work as follows:

5.1 Text retrieval

Once we have performed the OCR on the video and ASR on the audio, we will be able to perform a search of our user's query on the text transcripts. We will take the following steps during our text retrieval process:

- We will apply proter stemming to the target text sections and to the user query. This will remove suffixes from the text to improve the reliability of the system.
- Next we will remove stop words from the processed text. This will involve the usual stop words such as *to*, *and*, *it*; in addition to this we will be looking to remove some context specific stop words such as *uhh*, *umm* and other delaying noises in speech.
- We will then use one of either TF-IDF or Okapi BM25 to search for relevant sections of visual or speech text.

5.2 Architecture diagram

Insert High Level diagram of the system here.

5.3 Test effectiveness of the system

The two parameters we will need to evaluate for the success of our system will be effectiveness and efficiency.

We will use videos which we know the content of and search for sections which exist in the videos. We will evaluate the effectiveness based on the number of relevant sections found divided by the number of sections which were returned as relevant. This will give a percentage of precision for the system.

We will also evaluate the system using the A/B testing along with user feedback to find a level of relevancy which is apporopriate for highlighting sections in the system.

For efficiency we will look at time consumption of the system. We will aim to minimize our search time to be at least 60% or less of the time a user would normally spend manually clicking through a video for content. We hope to achieve this by serving up our search results as soon as we get them. We can deliver the results of the audio processing quicer than the video processing in some cases because it will be pre-processed for the video.

Notes

[1]ASR is the recognition of human speech and converting it to text.

[2]Put reference to: https://developers.google.com/youtube/v3/docs/captions

[3]https://cloud.google.com/speech/